НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»
Факультет інформатики та обчислювальної техніки
Кафедра автоматики та управління в технічних системах

«До захисту допущено»
Завідувач кафедри
_____ О.І. Ролік
«___»_____2019 р.

Дипломний проект

на здобуття ступеня бакалавра

з напряму підготовки 6.050103 «Програмна інженерія»

на тему: «Програмно-апаратна інфраструктура наземного робота для дослідження території»

Виконав:
студент IV курсу, групи ІТ-51
Бессмертний Роман Сергійович

_____

Керівник:
Доцент Катін П.Ю.

_____

Рецензент:
_____

_____

Засвідчую, що у цьому дипломному проекті немає запозичень з праць інших авторів без відповідних посилань.
Студент (-ка) _____

Київ – 2019 рік

Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
Факультет інформатики та обчислювальної техніки
Кафедра автоматики та управління в технічних системах

Рівень вищої освіти – перший (бакалаврський)
Напрям підготовки – 6.050103 «Програмна інженерія»

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ О.І. Ролік
«___»_____2019 р.

ЗАВДАННЯ
на дипломний проект студенту
Бессмертному Роману Сергійовичу

1.Тема проекту «Програмно-апаратна інфраструктура наземного робота для дослідження території», керівник проекту доцент Катін Павло Юрійович, затверджені наказом по університету від «___»_____ 2019 р. №_____

2. Термін подання студентом проекту _____

3. Вихідні дані до проекту

Мобільна навігаційна система на базі Orange Pi Zero Plus з WiFi-модулем, 2 двигуни постійного струму 3V-6B, сервопривід Futaba S3003, Raspberry PI камера, пластмасовий корпус. Операційна система Armbian, мови програмування Java, C та Javascript, середовище програмування IntelliJ Idea 2019, цільова платформа JVM, обрана для розробки технологія – Spring Framework, СУБД –SQLLite.

4. Зміст пояснювальної записки

1. Вступ 2. Огляд складових компонентів 3. Побудова діаграми розгортання апаратної інфраструктури 4. Побудова діаграми компонентів програмно-апаратної інфраструктури робота 5. Реалізація програмної частини інфраструктури робота

Додатки:
6. Код програми

5. Перелік графічного матеріалу

Діаграма розгортання, діаграма компонентів, діаграма послідовності передачі команд керування, діаграма послідовності передачі поточного відео, діаграма потоку даних

6. Дата видачі завдання _____

Календарний план

№ з/п	Назва етапів виконання дипломного проекту	Термін виконання етапів проекту	Примітка
1	Вибір тематичного напрямку та узгодження теми дипломного проекту	22.02.2019	
2	Аналіз теоретичних матеріалів та вивчення предметної області	15.04.2019	
3	Розробка технічного завдання, вибір методів та засобів реалізації задачі	24.04.2019	
4	Огляд існуючих рішень з тематики роботи	27.04.2019	
5	Розробка структури прототипу та проектування системи	06.05.2019	
6	Реалізація проекту	20.05.2019	
7	Налагодження та перевірка програми	23.05.2019	
8	Оформлення пояснювальної записки	03.06.2019	
9	Передзахист дипломного проекту	04.06.2019	
10	Доопрацювання пояснювальної записки та підготовка презентації	18.06.2019	
11	Захист дипломного проекту	20.06.2019	

Студент Р.С. Бессмертний

Керівник проекту П.Ю. Катін

Анотація

АНОТАЦІЯ

Бессмертний Р.С. Програмно-апаратна інфраструктура наземного робота для дослідження території. КПІ ім. Ігоря Сікорського, Київ, 2019.

Ключові слова: Мікроконтролери, програмна інфраструктура, діаграма розгортання, діаграма компонентів, робот дослідження території, Java, операційна система Armbian.

Основна частина документу викладена у пояснювальній записці, виконаній на 62 сторінках, та містить 20 рисунків та 8 таблиць. Також до його змісту входить 1 додаток.

Приведений огляд існуючих компонентів, з який можна зібрати наземного робота дослідження території. Було створено програмно апаратну інфраструктуру на основі мікрокомп'ютера Orange Pi Zero Plus з General Purpose Input/Output для керування такими периферійними пристроями, як двигуни постійного струму, та відеокамера.

Була розроблена система передачі команд керування, беручи до уваги вимоги безпеки використання. Була розроблена та оптимізована система передачі поточного відео. Була розглянута і теоретично розроблена система збереження даних, визначені основні потоки даних та сценарії їх обробки.

SUMMARY

Bessmertnyi R.S. "Software and hardware infrastructure of a ground robot used for territory exploration". Igor Sikorsky KPI, Kyiv, 2019.

Keywords: Microcontrollers, software infrastructure, deployment diagram, component diagram, robot used for territory exploration, Java, operating system Armbian.

The bulk of the document is outlined in the explanatory note, with 60 pages, and contains 20 figures and 8 tables. Also included in its content is 1 application.

An overview of existing components from which you can build a ground robot used for territory exploration is provided. A software hardware infrastructure was created based on the Orange Pi Zero Plus microcomputer with General Purpose Input / Output for controlling peripherals such as DC motors and a camcorder.

A system for transmitting control commands was developed, taking into account the safety requirements for use. A system for transmitting video stream was developed and optimized. The system of data storage was considered and theoretically developed, the main data streams and their processing scenarios were determined.

РЕЦЕНЗІЯ

на дипломний проект
на здобуття ступеня бакалавра,
виконаного на тему: «Програмно-апаратна інфраструктура наземного робота для дослідження території»
студентом Бессмертним Романом Сергійовичем.

Дана дипломна робота повністю відповідає затвердженій темі та дипломному завданню, реалізуючи поставлені вимоги до кожної складової програмно-апаратної інфраструктури. Тема розробки роботів дослідження території є актуальною в наш час, а предметна область та способи вирішення таких задач відповідають науковій тематиці кафедри.

Був проведений вичерпний аналіз існуючих компонентів, протоколів обміну даними та існуючих програмних рішенн. Моделювання апаратної частини за допомогою діаграми розгортання, а програмної за допомгою діаграми компонентів допомагає краще зрозуміти всі складові та деталі робота дослідження території.

При конструюванні ПЗ були використані сучасні технології та методики: Spring Framework, Video4Linux4Java, WiringPi, Sqlite. В контексті використання повноцінної операційної системи дані технології дають реальну можливість істотно зменшити вартість розробки програмної інфраструктури робототехніки та інших вбудованих систем.

З недоліків розробленого продукту можу виділити передачу кожного кадру окремим зображенням, що не дозволяє використовувати сучасні технології стиснення відеоданних і створює зайве навантаження на мережу. З іншої сторони, система обробки команд керування дозволяє забезпечити високу надійність і безпеку використання, що є істотною перевагою в контексті використання дешевших компонентів апаратної інфраструктури.

В цілому проект відповідає необхідним вимогам та заслуговує оцінки «_____», а його автор — Бессмертний Роман Сергійович — присудження першого ступеня вищої освіти бакалавр за напрямом підготовки 6.050103 «Програмна інженерія» і присвоєння кваліфікації бакалавр програмної інженерії.

Рецензент

ВІДГУК

керівника дипломного проекту
на здобуття ступеня бакалавра,
виконаного на тему: «Програмно-апаратна інфраструктура наземного робота для дослідження території»
студентом Бессмертним Романом Сергійовичем.

Дипломна робота є самостійним, змістовним та актуальним дослідженням. Була досягнута відповідність поставленому завданню.

Дипломна робота порушує питання використання багатоядерних високопотужних плат мікрокомпютерів. Був проведений вичерпний аналіз існуючих компонентів, протоколів обміну даними та існуючих програмних рішеннь. Для створення програмно-апаратної інфраструктури були досліджені нові технології, взаємодія яких на цей час ще не є вивченою.

Студент показав готовність до прийняття сучасних рішень, уміння самостійно аналізувати необхідні літературні джерела, приймати правильні інженерні та наукові рішення, застосовувати сучасні системні та інформаційні технології. Було проведене фізичне моделювання, на основі якого були поставлені експерименти. Результати експериментів були оброблені, проаналізовані та дозволили прийняти оптимальні інженерні рішення. До найбільш важливих теоретичних і практичних результатів потрібно віднести дослідження можливості використання високопотужних плат мікрокомпютерів та впровадження новітніх програмних технологій на їх основі. Ці дослідження стали основою публікації в науково-технічному журналі «Стандартизація, сертифікація, якість».

Автор проекту, Бессмертний Роман Сергійович, заслуговує присудження першого ступеня вищої освіти бакалавр за напрямом підготовки 6.050103 «Програмна інженерія» і присвоєння кваліфікації бакалавр програмної інженерії.

Керівник дипломного проекту

к.т.н., доцент Катін П. Ю

Номер рядка	Формат	Позначення	Найменування	Кіл. листів	№ екз.	Примітка
1			Документація загальна			
2						
3			Знову розроблена			
4						
5	А4	ІТ51.020БАК.002 ПЗ	Пояснювальна записка			
6						
7	А3	ІТ51.020БАК.003 Д1	Діаграма розгортання	1		
8						
9	А3	ІТ51.020БАК.004 Д2	Діаграма компонентів	1		
10						
11	А3	ІТ51.020БАК.005 Д3	Діаграма послідовності команд керування	1		
12						
13	А3	ІТ51.020БАК.006 Д4	Діаграма послідовності поточного відео	1		
14						
15	А3	ІТ51.020БАК.007 Д5	Діаграма потоку даних	1		
16						
17						
18						
19						
20						
21						
22						
23						
24						

					ІТ51.02.0БАК.001 ТП		
Змн.	Арк.	№ докум.	Підпис	Дата			
Розроб.		Бессмертний Р.С.			Програмно-апаратна інфраструктура наземного робота для дослідження території. Відомість технічного проекту	Літ.	Арк. 1 / Акрушів 61
Перевір.		Катін П. Ю.					
Реценз.						КПІ ім. Ігоря Сікорського ФІОТ, гр. ІТ-51	
Н. Контр.		Шинкевич М.К.					
Затверд.							

Пояснювальна записка
до дипломного проекту
на тему: «Програмно-апаратна інфраструктура наземного робота для дослідження території»

Київ – 2019 рік

ЗМІСТ

ВСТУП

Роботи - автоматичні системи, призначені для відтворення рухових та інтелектуальних функцій людини. Від традиційних автоматів відрізняються більшою універсальністю і здатністю до адаптації для виконання різних завдань, в тому числі і за мінливих обставин.

Ідея побудови доступних у промисловості, побуту, торгівлі роботів далеко не нова, але саме останнє десятиліття принесло нові технології та методи, що дозволяють реалізувати доступні, потужні та надійні системи. Для реалізації таких систем роботів потрібно програмно-апаратна інфраструктура(ПАІ) наземного робота для дослідження території(НРДТ). Дані НРДТ зможуть допомагати виконувати будь-які види робіт – від найпростішої рутини до складних процесів, небезпечних для життя людини.

В основі таких НРДТ лежать мікроконтролерні системи різного ступеня складності, що являють основу ПАІ. Найбільш розповсюдження отримали мікроконтролерні системи середньої і малої потужності 8-ми і 16-ти розрядні [1-2]. Активно іде розвиток багатоядерних високопотужних мікроконтролерів[3-4], що можуть стати основою ПАІ для НРДТ. Вартість таких ПАІ наближається до вартості розповсюджених контролерів середньої і малої потужності[1-2].

Перевагою даного типу ПАІ полягає в можливості використання повноцінної операційної системи. Це дозволяє використовувати всі сучасні технології програмування, у тому числі мову програмування Java і звязані з неї фрейморки. Це значно зменшує вартість розробки програмного забезпечення. Отже кінцева вартість розробки робототехніки на базі ПАІ на основі високопотужних мікроконтролерів може бути меншою вартості аналогічного контролера на базі мікроконтролерів середньої і малої потужності.

В даній галузі потрібно визначити, що на теперішній час недостатньо публікацій, що досліджують програмно-апаратну інфраструктуру робототехніки на базі плат високопотужних мікрокомпютерів. При цьому прикладні і інженерні рішення існують і активно використовуються на практиці.

Метою даної роботи є формулювання загальноприйнятого в ІТ опису елементів архітектури ПАІ на основі мікрокомпютерів на прикладі мобільної платформи для дослідження території.

Для досягнення поставленої мети необхідно:

- побудувати прототип апаратної частини ПАІ;
- побудувати прототип програмної частини ПАІ на основі Armbian;
- розробити прототип програми для реалізації основи НРДТ.

1 АНАЛІЗ СКЛАДОВИХ КОМПОНЕНТІВ

Для тестування та оцінки можливості побудови прототипу ПАІ для НРДТ проаналізуємо доступні за вартістю рішення на основі мікроконтролерів cortex A а саме лінійку мікрокомпютерів Orange Pi. Для цього в розділі проаналізовано можливість розробки програмної і апаратної частини ПАІ. Це реалізовано на базі системи передачі команд керування, і систему передачі поточного відео. У цьому розділі буде детально розглянути компоненти прототипу ПАІ для НРДТ та можливості їх використання.

1.1 Аналіз існуючих плат мікрокомпютерів.

Orange Pi Zero Plus[5] є платою мікомпютера(ПМК), який містить мінімально необхідну кількість компонентів, які забезпечують його функціонування.

ПМК містить на собі роз’єми введення та виведення для створення апаратної інфраструктури. В даний час компанія продає голу ПМК - корпус не входить у комплект. Що ж стосується програмного забезпечення, в даний час є три операційних системи на основі Linux, підтримуваних Orange Pi Zero Plus. Що стосується специфікацій, то Orange Pi Zero Plus є пристроєм, розміром 48 x 46 мм. Система включає в себе 64-бітний процесор Allwinner H5 Quad-core Cortex-A53, що працює на частоті 1.2 ГГц, а також графічний процесор Hexa-core Mali450. Вона має 512 МБ оперативної пам'яті DDR3. Orange Pi отримує енергію від зарядного пристрою Micro USB 5V 2A AC, і вимагає стабільної напруги для надійної роботи. У той час як ARM процесор забезпечує реальну продуктивність, Hexa-core Mali450 GPU є потужним графічним ядром, здатним до апаратного декодування декількох форматів відео високої чіткості.

В даній роботі була використана модель Orange Pi Zero Plus. Плата оснащена лише композитним відеовиходом, але відеовихід не впливає на створення програмно-апаратної інфраструктури наземного робота дослідження території. Плата має один порти USB 2.0, порт 10/100/1000 Ethernet, слот для SD-

карти, GPIO (General Purpose I / O) роз'єм і аналоговий аудіо вихід (3,5 мм роз'єм для навушників).

Далі розкриті технічні характеристики ПМК на базі OrangePi Zero Plus, а саме [5]:

– процесор 64-бітний SoC Allwinner H5 Quad-core Cortex-A53;

– графічний процесор Mali450MP4;

– оперативна пам’ять 512Mb DDR3 synchronous dynamic random-access memory (включаючи GPU);

– вбудована пам'ять 8GB embedded MultiMediaCard;

– можливість підтримки карт пам'яті microSD (об'ємом до 64GB);

– аудіо- і відеовивід через HDMI з підтримкою Consumer Electronics Control;

– вбудований WiFi 802.11 b/g/n + Bluetooth 4.0 Low Energy;

– microUSB 2.0 On-The-Go x1;

– Camera Serial Interface;

– налагоджувальний порт;

– 26 пінів General Purpose Input-Output, сумісних з Raspberry Pi B +;

– світлодіоди стану і активності;

– живлення від microUSB роз'єму;

– розміри 48 x 46 mm;

– вага 20g.

Основною перевагою ПМК Orange Pi є те, що цей пристрій є вдалою комбінацією невеликого розміру комп'ютера і доступної ціни. Ентузіасти часто використовують цю та аналогічні ПМК в якості дешевого ПК для домашнього кінотеатру або неосновного робочого комп’ютера з низьким рівнем енергоспоживання. Невеликий розмір робить дозволяє легко приховати комп'ютер, що може бути встановлений позаду дисплея за потреби. Він також може бути використаний в нішевих додатках, таких як цифрові вивіски або як мікрокомп’ютер для визначення проблем на основній робочій машині.

Розташування описаних компонентів зображено на принциповій схемі(рис. 1.2).

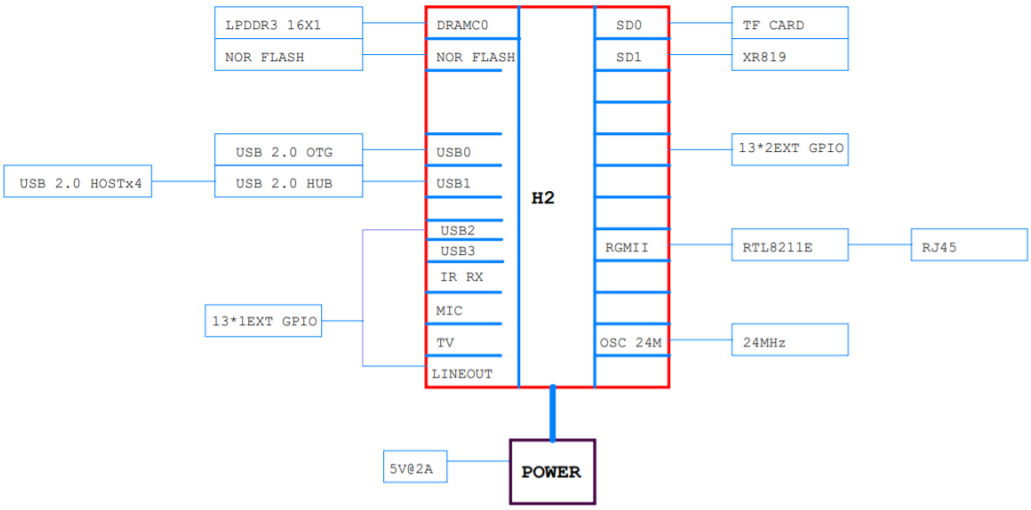

Рисунок 1.2 – Схема компонентів Orange Pi Zero Plus[6]

Orange Pi це не єдиний маленький пристрій в своєму роді - два інших відомих приклади в ентузіастів співтовариства є Arduino і Raspberry Pi. Хоча аналогічні системи дуже схожі за призначенням, Orange Pi має суттєві відмінності від цих систем. З апаратної точки зору, Raspberry Pi базується навколо ARM системи, у якої дуже закритий вихідний код. З іншого боку, системи Orange Pi засновані на апаратних засобах з повністю відкритим вихідним кодом. Плати Arduino є ще більш несхожі в зв'язку з використанням 8-бітних і 16-бітних Atmel плат мікроконтролерів.

Найбільша різниця між чимось на зразок Arduino і Orange Pi полягає в цілях використання. Arduino призначений для інтеграції в більші машини або електроніку. Orange Pi ж призначений для використання в якості кінцевого продукту і для роботи в якості традиційного настільного комп'ютера. Були розглянуті основні особливості ПМК, зображеного на рис. 1.3.

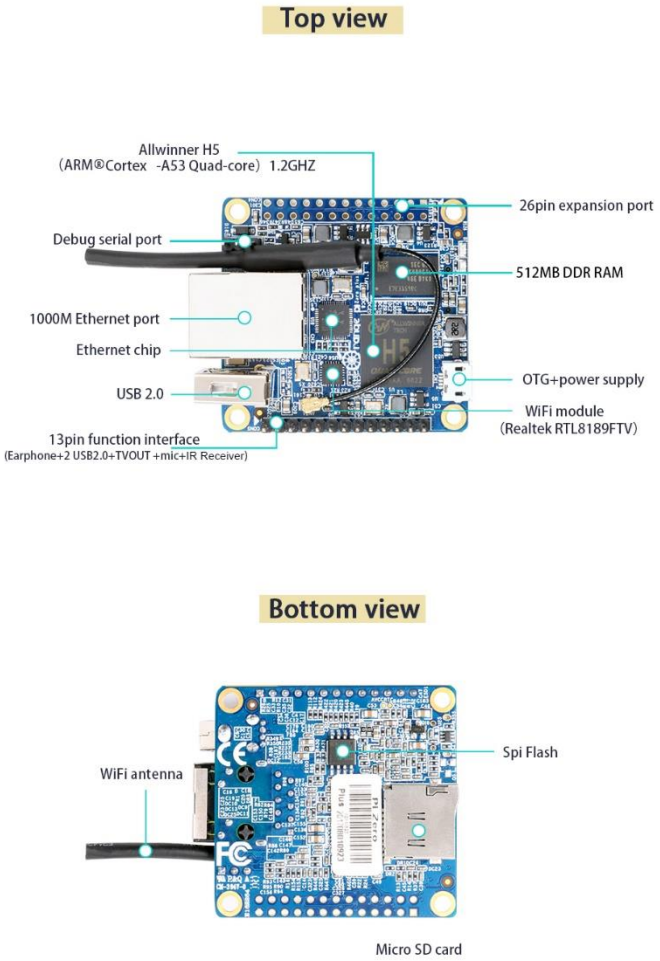

Рисунок 1.2 – Зовнішній вигляд плати мікроконтролера Orange Pi Zero Plus[7]

Однією з особливостей Orange Pi є ряд GPIO (General Purpose Input/Output) - шпильки уздовж краю дошки. Ці контакти є фізичним інтерфейсом між Pi і зовнішнім світом. На найпростішому рівні, вони можуть розглядатись як перемикачі, що можна увімкнути або вимкнути або які сама Raspberry Pi може увімкнути або вимкнути . Сімнадцять з 26 контактів GPIO є контактами введення/виведення; інші контакти - електричні або контакти заземлення. Схема контактів GPIO з їх призначенням зображена на рис 1.3.

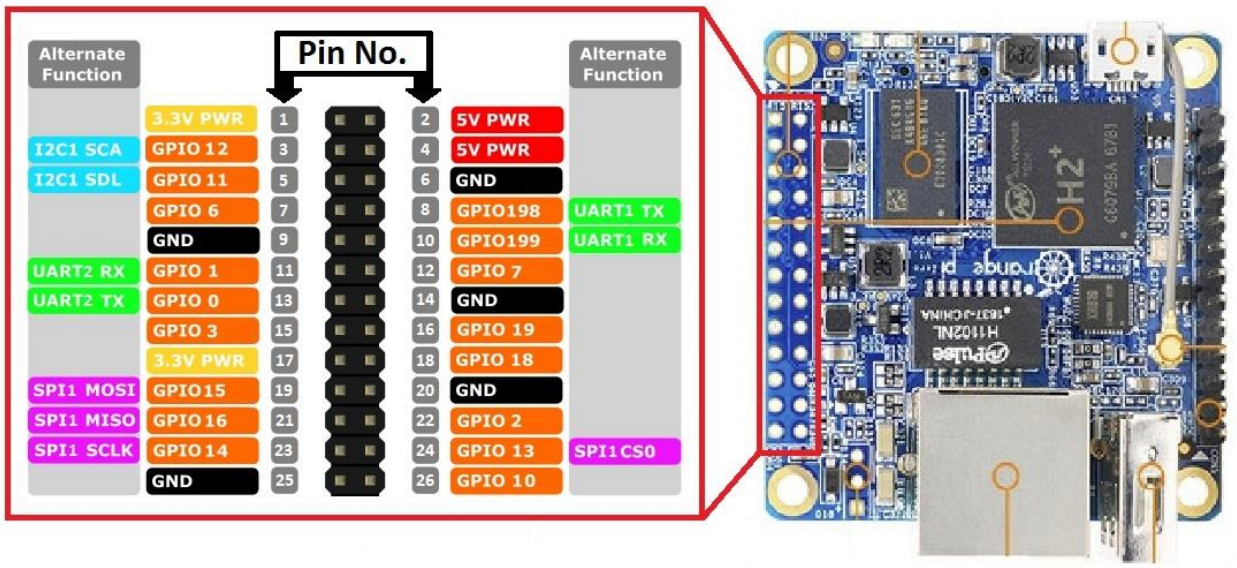

Рисунок 1.3 - Схема GPIO з фізичною нумерацією контактів[8]

GPIO під'єднане до обчислювальної системи процесора Cortex-A53 трьома контактами. Кожен з 3-х контактів має свій власний VDD вхідний контакт. На Orange Pi всі контакти GPIO живляться від 3.3В. Підключення GPIO до напруги вище, ніж 3.3V, може зруйнувати блок GPIO. Коли ми використовуємо GPIO контакти в якості вихідного сигналу, Orange Pi замінює перемикач і батареї живлення. Кожен контакт можна увімкнути або вимкнути, тобто встановити високий або низький сигнал. Коли на контакті високий сигнал - виводить 3,3 вольт; коли на контакті низький сигнал - він вимкнений.

Інші ПКМ даного класу мають подібні характеристики і не поступаються потужністю і функціональністю. Невисока вартість, $15 на момент написання статі, наявність безкоштовної ОС "Armbian", сумісність даної ОС з технологіями JAVA[13-22] і наявність безкоштовних, надійних бібліотек для виконання поставленої задачі дозволяють, крім відомих методів контролю якості виробництва харчових продуктів шляхом випарювання, забезпечити контроль продукції за зовнішнім виглядом.

На теперішній час провадяться роботи із використання ПМК (наприклад OrangePi) в якості контролера виробництва. Існує бізнес з застосування ПМК для реалізації промислових контролерів управління технологічними процесами [12]. Прикладом є лінійка продуктів ModBerry, що використовує ПМК OrangePi (або

аналоги) для управління промисловими виробництвами. Також реалізована серія контролерів ModBerry M300 O1 / O2, що утворена двома модульними ПМК. Відомо багато аналогічних рішень. Крім вищеописаних характеристик ця система реалізує провідні та бездротові інтерфейси, наприклад HDMI, Wi-Fi, Bluetooth 4.0.

1.2 Бездротовий зв'язок.

Вбудовані пристрої та системи історично підключалися дротяними інтерфейсами для обміну даними та обслуговування. Для реалізації прототипу НРДТ дротяний зв'язок не є оптимальним. Згідно з предметною областю робот повинен мати можливість передавати відеодані, бажано в реальному часі. Робоча відстань такого зв’язку повинна бути істотною. Мобільна платформа має мати можливість дослідження території, що обмежує мінімальну відстань зв’язку розмірами досліджуваної території. Зв'язок на великих відстанях є бажаним, проте не обов’язковим.

Хоча нові технології і інтерфейси зробили вбудовування бездротового зв'язку більш практичним і економічним, недоліком є те, що сьогодні існує широкий і заплутаний набір доступних протоколів, що розрізняються діапазоном і швидкістю передачі даних. Це ускладнює для розробників зробити правильний вибір для конкретного додатка. Далі порівнюються і підсумовуються десять варіантів бездротових мереж для вбудованих систем.

1.2.1. Критерії оцінки бездротового інтерфейсу

Діапазон, вартість і енергоспоживання є, мабуть, найбільш важливими критеріями для більшості вбудованих систем. Що стосується діапазону, параметри бездротового зв'язку широко варіюються:

– NFC діє тільки на кілька сантиметрів;

– Bluetooth і Zigbee розраховані на кілька метрів, використовуючи надзвичайно низьку потужність;

– приймач Wi-Fi на основі 802.11 мають діапазон, який обчислюється в сотнях метрів;

– вузькосмуговий IoT (NB-IoT) використовує ліцензовану стільниковий інфраструктуру для перенесення бездротової інформації на багато кілометрів;

– LoRaWAN і Sigfox - малопотужні бездротові пристрої великої дальності для пристроїв IoT, які також охоплюють багато кілометрів, але працюють в неліцензованих діапазонах;

На рисунку 1.4 наочно видно функціональне співвідношення розглянутих бездротових протоколів.

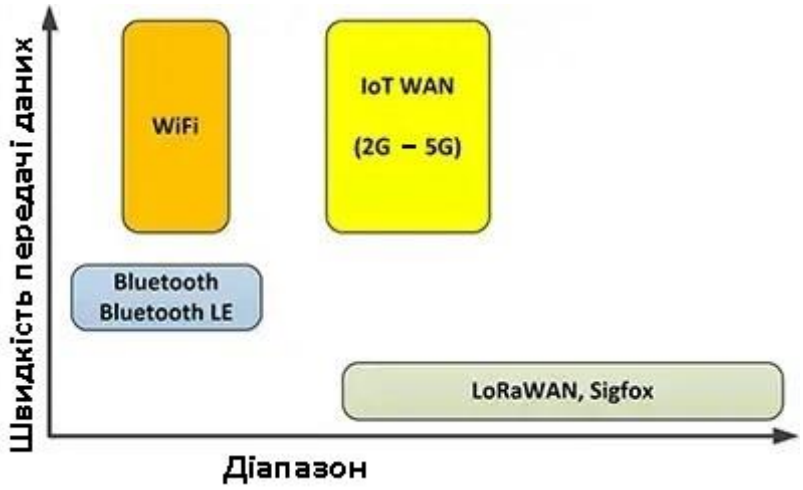

Рисунок 1.4 Функціональне співвідношення бездротових протоколів

На додаток до діапазону, вартості та споживання енергії є два додаткових критерії. По-перше, потрібно зрозуміти, чи потребує додаток вбудований процесор додатків. Деякі бездротові модулі емулюють роботу і використовують ті ж засоби розробки, що і популярні плати розробки, такі як Arduino Uno. Інші мають свої власні архітектури і власні екосистеми розвитку. У інших немає ніякої вбудованої можливості обробки взагалі.

Якщо бездротовий модуль буде реалізовувати зв'язок тільки для хост-процесора, то важливим є інтерфейс між хост-процесором і бездротовим модулем. Тут є багато варіантів, включаючи послідовні протоколи, такі як I2C, SPI або UART. Інша можливість - це лінії введення / виведення Arduino - багато модулі доступні як Шілд Arduino. Однак ці повільні послідовні інтерфейси і лінії

введення / виведення Arduino не підтримуватимуть більш високі швидкості передачі даних. Більш швидка швидкість передачі даних вимагає набагато більш швидких інтерфейсів, таких як PCIe, наприклад.

У таблиці 1.1 в алфавітному порядку перераховані десять загальних варіантів вибору і основних критеріїв вибору для різних протоколів бездротового зв'язку для вбудованих проектів.

Таблиця 1.1 Порівняння протоколів бездротового зв’язку.

Стандарт	Потужність	Діапазон	Швидкість
Bluetooth	Середн.	1 до 100 м	1 to 3 Mbps
Bluetooth LE	Низьк.	>100 м	125 kbps to 2 Mbps
LoRaWAN	Низьк.	10 км	0.3 to 50 kbps
NB IoT	Низьк.	<35 км	20 kbps to 5 Mbps
NFC	Низьк.	<10 см	106 to 424 kbps
Sigfox	Низьк.	3 до 50 км	100 to 600 kbps
6LoWPAN	Низьк.	100 м	0 to 250 kbps
802.11 / Wi-Fi:	Середн.	100 м – 10 км	10 to 100+ Mbps
802.15 / Zigbee	Низьк.	10 до 100 м	20 to 250 kbps
Z-Wave	Низьк.	15 до 150	9.6 to 40 kbps

Деякі з цих бездротових протоколів, таких як Wi-Fi, Bluetooth, Bluetooth з низьким енергоспоживанням (LE) і NFC, вже широко використовуються в мобільних телефонах і портативних комп'ютерах. Нижче наведено короткі зведення по кожному з бездротових стандартів, перерахованих в таблиці вище.

Спочатку розроблений для бездротового зв'язку супутніх пристроїв з мобільними телефонами, Bluetooth став корисним бездротовим протоколом для додатків з малою потужністю, які вимагають відносно короткою і помірної смуги пропускання даних від 1 до 3 мегабіт в секунду (Мбіт / с). Через поширеності протоколу модулі Bluetooth RF відносно легко інтегруються у вбудований додаток.

Bluetooth LE значно знижує енергоспоживання і витрати в порівнянні з класичним Bluetooth, підтримуючи аналогічний діапазон зв'язку. Він націлений на нові додатки в галузі охорони здоров'я, фітнесу, радіомаяків, безпеки і домашніх розваг.

LoRaWAN призначений для бездротових пристроїв з батарейним харчуванням в регіональній, національній або глобальної мережі, LoRaWAN націлений на ключові вимоги IoT для забезпечення безпечної, малопотужної, двобічної зв'язку з мобільністю і послугами локалізації на широкій території. Специфікація LoRaWAN - це рівень управління доступом до середовища (MAC), який може бути накладено на різні протоколи фізичного рівня (PHY) від супутникових мереж, таких як Globalsat, до наземних публічних і приватних мереж. LoRaWAN забезпечує безшовну і довгострокову сумісність між пристроями IoT без необхідності підтримки локальної мережі.

Вузькосмуговий (Narrowband) IoT був розроблений для підключення широкого спектру пристроїв і надання послуг з використанням стільникових мереж зв'язку, Narrowband IoT (NB-IoT) є одним з ряду технологій Mobile IoT (MIoT), стандартизованих в рамках проекту партнерства третього покоління (3GPP). NB-IoT розгортається «внутріполосное» в стільниковому спектрі, виділеному для стільникових мереж 4G LTE, використовуючи блоки ресурсів в нормальній несучої LTE або в невикористовуваних блоках ресурсів в захисній смузі несучої LTE.

Для портативних пристроїв, таких як мобільні телефони, NFC надає стандартизований набір протоколів зв'язку, які дозволяють двом електронним пристроям спілкуватися в безпосередній близькості (зазвичай менше 10 см), тому це строго короткий з'єднання. Він часто використовується для фінансових транзакцій, таких як безконтактні платіжні системи та електронні мобільні квитки. Через коротке діапазону NFC одне з двох комунікаційних пристроїв NFC зазвичай є портативним. В іншому випадку проста пара проводів зазвичай забезпечує більш дешеву і просту зв'язок.

Малопотужні об'єкти, такі як лічильники електроенергії або смарт-карти, які необхідно періодично перевіряти і які повинні працювати від батареї протягом багатьох років або навіть десятиліть, можуть використовувати власний радиоинтерфейс Sigfox, щоб іноді відправляти невеликі обсяги даних в хмару.

6LoWPAN це абревіатура «IPv6 over Low-Power Wireless Personal Area Networks», це засноване на ідеї, що інтернет-протокол (IP) може і повинен бути застосовний навіть до найменших пристроїв. Протокол 6LoWPAN дозволяє малопотужним пристроїв з обмеженими можливостями обробки брати участь в мережах IoT, визначаючи механізми, які дозволяють відправляти і приймати пакети IPv6 по радиосетям на основі менш складних рівнів PHY і MAC IEEE 802.15.4 (який також служить в якості основи для низькочастотних мереж Zigbee і інших).

802.11 / Wi-Fi це повсюдний, швидкий і з власної підтримкою IP, радиоинтерфейс. Wi-Fi відносно легко інтегрується у вбудовану систему для безпосереднього підключення пристрою до IoT.

802.15 / Zigbee це стандарт IEEE 802.15.4 визначає PHY і MAC для бездротових персональних мереж з низькою швидкістю передачі даних (WPAN). Zigbee ґрунтується на стандарті 802.15.4 з бездротовим протоколом, призначеним для створення середніх або великих пористих мереж, які пов'язують датчики і контролери.

Z-Wave був розроблений як простий у використанні низькоскоростний протокол бездротового зв'язку, який дозволяє різним пристроям домашньої електроніки взаємодіяти з використанням надійного бездротового протоколу з низьким енергоспоживанням, який легко передає інформацію через стіни, підлоги і шафи. Z-Wave - це пропріетарний протокол, розроблений одним консорціумом і вимагає ліцензії на використання. У Z-Wave Alliance налічується понад 700 компаній-членів, що пропонують понад 2400 бездротових, «інтелектуальних» продуктів, таких як побутові прилади, віконні штори, термостати і домашнє освітлення.

Наявність вбудованого Wi-Fi адаптера, достатня швидкість для передачі потокового відео, а також можливість і простота використання існуючих веб-технологій обумовлюють вибір протоколу Wi-Fi як найдоцільнішого для створення апаратної інфраструктури мобільної платформи.

Враховуючи необхідність передачі потокового відео, на теперішній час протокол IEEE 802.11, тобто Wi-Fi, виявляється найдоцільнішим.

1.2.2. Wi-Fi адаптер RTL8189FTV

У даній роботі був використаний вбудований адаптер RTL8189FTV QFN24. Це один з найбільш якісних WiFi модулей, який підтримується всіма операційними системами на базі Linux або Windows. Може працювати в режимах IEEE 802.11b, IEEE 802.11g и IEEE 802.11n. Його принципова схема наведена на рисунку 1.7.

Для підключення до мережі через командний рядок було написано утиліти для пошуку всіх доступних мереж та швидкого підключення до потрібної з введенням паролю.

1.3 Камера

Основним пристроєм, який буде використовуватись для локалізації робота у просторі, буде камера. Зображення будуть оброблятись та на основні цих даних робот-автомобіль буде робити припущення, де саме він знаходиться.

У даній роботі була використана 1.3-мегапіксельна камера Gear Head WC740i-CP10 на базі сенсора OV5647. Камера сумісна з усіма версіями Orange Pi. Вона прикріплюється до Orange Pi через стандартний роз’єм USP 2.0.

Рисунок 1.6 - 1.3-мегапіксельна камера Gear Head WC740i-CP10[9]

Камера невелика у розмірах: 25 х 24 х 9 мм. Розширення фото, отриманих з камери - 640x480 пікселів. Підтримувані формати відео - 1080p з швидкістю 30 кадрів за секунду(fps), 720p з 60 fps і 480p з 90 fps. Фокусна відстань дорівнює 8 мм.

Для отримання зображення у алгоритмі використана обгортка на мові програмування Java, що буде детально розглянута далі.

1.4 Модуль L298N H-bridge

Модуль L298N H-bridge[11], зображений на рисунку 1.7, дуже простий у використанні - він підтримує одночасно два двигуни для підключення і може керувати ними одночасно. На входи ми подаємо виходи двигунів, на виходи - 4 керуючі сигнали для двигунів, які підуть на I/O контакти GPIO, напруга на модуль подається зі стороннього джерела струму, так як потужності Orange Pi не вистачає для живлення двигунів.

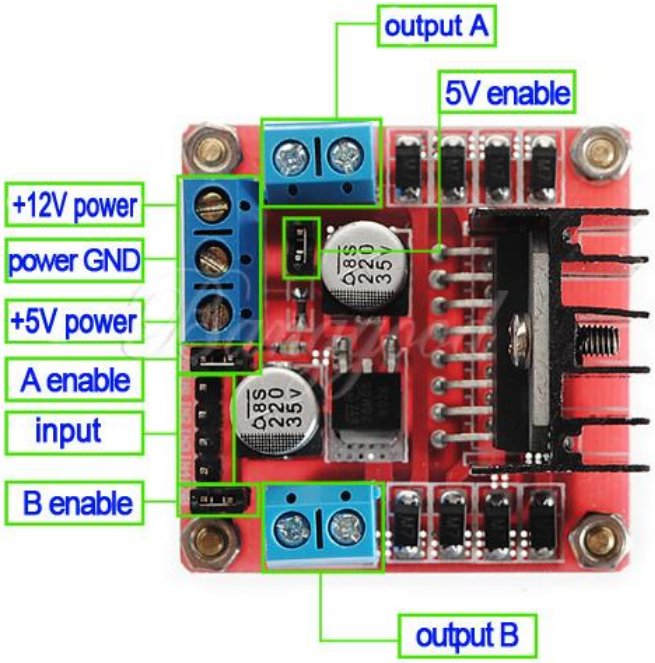

Рисунок 1.7 - Модуль L298N H-bridge 24[10]

Кожен HG7881 (L298N) чіп здатний керувати одним двигуном постійного струму за допомогою двох цифрових входів управління. Один вхід використовується для вибору напрямку обертання двигуна в той час як інший використовується для управління швидкістю обертання двигуна. Швидкість регулюється за допомогою широтноімпульсної модуляції(ШІМ). Пристрій має свою таблицею істинності, яка визначає результат подачі сигналів на його входи. Знайти її можна в табл. 2.1.

Таблиця 2.1 - L298N Таблиця істинності

Вхід		Вихід		
IA	IB	OA	OB	Опис
0	0	0	0	Вимкнений
1	0	1	0	Вперед
0	1	0	1	Назад
1	1	1	1	Вимкнений

Варто звернути увагу, що фактичний напрямок "вперед" і "назад" залежить від того, як двигуни змонтовані і підключені. Завжди можна змінити напрямок двигуна шляхом зміни його проводки. L298N використовує двоканальний

модуль двигуна. Кожен канальний драйвер призначений для управління одним двигуном, так що наявність двох означає, що цей модуль може управляти двома двигунами незалежно один від одного. Кожен канал двигуна використовує ту ж таблицю істинності, як описано вище. Кожен набір гвинтових клем використовується для підключення двигуна.

Згідно з таблицею 2.2 рекомендується використовувати вхід 1A для управління швидкістю обертання кожного двигуна і вхідний 1B для керування напрямком [15].

Таблиця 2.2 - Таблиця з’єднань

Вхід	Опис
B-IA	Двигун B вхід A
B-IB	Двигун B вхід B
GND	«Земля»
VCC	Операційна напруга 2.5-12B
A-IA	Двигун A вхід A
A-IB	Двигун A вхід B

1.5 Шасі

На даний час на ринку існує багато шасі для мобільних платформ. Зазвичай вони складаються з структурних деталей та двигунів. Структурні деталі як правило виготовлені з акрилу та оснащені отворами для монтажу апаратної інфраструктури.

Можлива переробка готових продуктів, наприклад радіокерованих іграшок, проте складність монтажу електроніки та низька ціна готових шасі роблять цей вибір не оптимальним.

Були розглянуті різні типи шасі, наявних на ринку[17]. Далі наведені переваги та недоліки кожного.

1.5.1. Колісні роботи

Колеса, безумовно, є найпопулярнішим методом забезпечення мобільності робота і використовуються для пересування багатьох різних роботів і робототехнічних платформ різного розміру. Колеса можуть бути практично будь-якого розміру, від декількох сантиметрів до 30 см і більше. Настільні роботи, як правило, мають найменші колеса, зазвичай менше 5 см в діаметрі. Роботи можуть мати практично будь-яку кількість коліс, хоча 3 і 4 є найбільш поширеними. Зазвичай триколісний робот використовує два колеса і заклинач на одному кінці. Більш складні двоколісних роботи можуть використовувати гіроскопічну стабілізацію. В більшості випадків колісний робот використовує поворот зсувом (подібно до танка). Для рульового управління з рейковим механізмом, таким як той, що знаходиться на автомобілі, потрібні дуже багато деталей, а його складність і вартість переважають більшість його переваг. Чотири і шести колісні роботи мають перевагу використання декількох приводних двигунів (один підключений до кожного колеса), що зменшує ковзання. Крім того, всенаправлені колеса або колеса mecanum, які використовуються належним чином, можуть надати роботові значні переваги мобільності.

Переваги колісних роботів:

– зазвичай дешевші в порівнянні з іншими методами;

– простий дизайн і будівництво;

– багатий вибір;

– шість коліс або більше можна порівняти з гусеничною системою;

– оптимальний вибір для початківців.

Недоліки колісних роботів:

– може втратити тягу (ковзання);

– мала площа контакту (лише невеликий прямокутник або лінія під кожним колесом контактує з землею).

1.5.2. Гусеничні роботи

Гусениці - це те, що використовують танки. Хоча гусениці не забезпечують додану "силу" (крутний момент), вони зменшують ковзання і більш рівномірно розподіляють вагу робота, роблячи їх корисними для сипучих поверхонь, таких як пісок і гравій. Крім того, система треків з деякою гнучкістю може краще відповідати вибоїстій поверхні.

Переваги гусеничних роботів:

– постійний контакт із землею запобігає ковзанню, що може статися з колесами;

– рівномірно розподілена вага допомагає вашому роботові вирішувати різноманітні поверхні;

– може використовуватися для значного збільшення дорожнього просвіту робота без використання більшого привідного колеса.

Недоліки гусеничних роботів:

– при повороті є бокова сила, яка діє на землю; це може призвести до пошкодження поверхні, на якій використовується робот, і призвести до зносу доріжок;

– немає багато різних варіантів гусениць (робот зазвичай будується навколо гусениць);

– зірочка приводу може значно обмежити кількість двигунів, які можна використовувати;

– підвищена механічна складність (розміщення холостого ходу і номер, кількість посилань) і кількість з’єднань;

1.5.3. Маніпулятори

Все більше роботів використовують ноги для мобільності. Ноги часто кращі для роботів, які повинні переміщатися на дуже нерівній місцевості. Більшість аматорських роботів розроблені з шести ногами, які дозволяють

роботові статично збалансовані (збалансовані у будь-який час на 3 ногах); Роботи з меншою кількістю ніжок важче врівноважити. Останні потребують «динамічної стабільності», тобто, якщо робот зупиняється посередині шагу, він може впасти. Дослідники експериментували з моноподними (одноногими «стрибками») конструкціями, хоча дві, чотири або шість ніг є найбільш популярними.

Переваги роботів з ногами:

– ближче до органічного або природного руху;

– потенційно можуть подолати великі перешкоди і пересуватися по дуже нерівній місцевості.

Недоліки роботів з ногами:

– збільшена механічна, електронна і програмна складність (не найпростіший спосіб потрапити в робототехніку);

– менший розмір батареї, незважаючи на підвищені вимоги до потужності;

– більш високі витрати на будівництво.

1.5.4. Опис та обґрунтування обраного шасі

Враховуючи розглянуті варіанти було обрана триколісна платформа. Триколісна платформа використовує два колеса і заклинач на одному кінці. В даному випадку шасі використовує поворот зсувом (подібно до танка). Причинами такого вибору стала низька ціна та простота роботи. У ході роботи був виявлений недолік у вигляді низької прохідності та проковзування на нерівній поверхні, наприклад килимах. Ці недоліки не були суттєвими для результатів дослідницької роботи, і не переважують значні переваги обраного шасі.

Шасі виготовлене китайської компанію Elecrow. Воно складається з пластикових компонентів, що з’єднуються за допомогою болтів, та трьох коліс, виготовлених з гуми та пластику.

1.6 Двигуни

Прототип НРДТ рухає колесами за допомогою двох двигунів. Задній привід побудований на двох двигунах постійного струму з робочою напругою від 3 до 6 Вольт.

Швидкість обертання двигуна залежить від величини напруги, що ми подаємо на вхід. Сам двигуни має лише два входи - VCC та GND. Тобто, автоматично, коли ми подаємо напругу, двигун буде працювати. У даній роботі керування двигунами має здійснюватися програмно з Orange PI, тож як перемикач напруги використовується модуль L298N H-bridge [11].

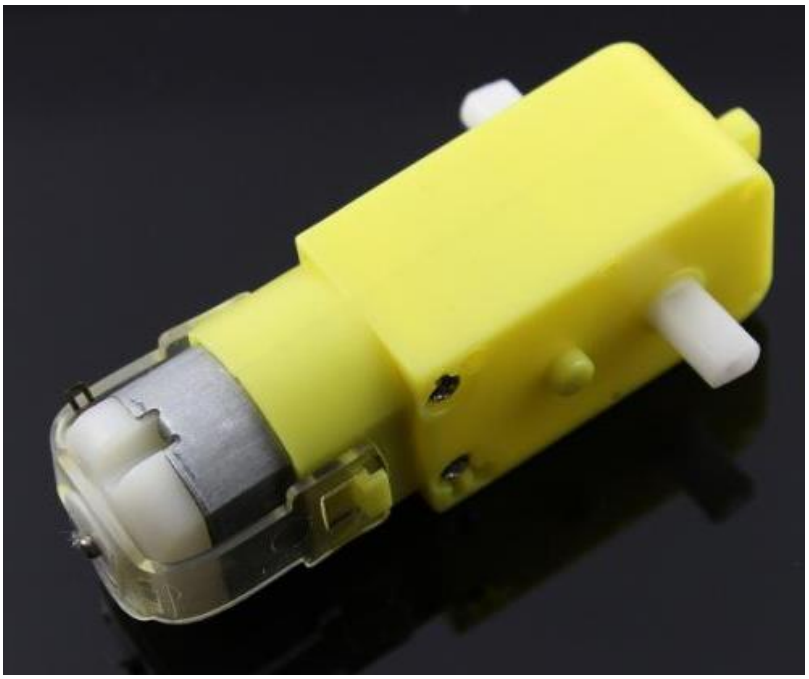

Рисунок 1.9 - Двигун постійного струму[14]

1.7 Джерело живлення

1.7.1. Види батарей, що використовуються в робототехніці

На ринку є багато різних типів акумуляторів[15], однак для простоти ми розділимо їх на дві групи.

Батареї що відмінно підходять для роботів:

– Li-Ion літій-іонний акумулятор;

– Li-Poly-літій-полімерні батареї;

– NiMH - нікель-металогідридна батарея.

І батареї що не підходять для використання в робототехніці, наведені далі.

Свинцево-кислотні акумулятори - всі типи (включаючи VRLA, SLA, гель або AGM) погано переносять цикли зарядки-розрядки і працюють набагато краще, як резервний блок живлення для стаціонарних додатків. Вони також мають низьку продуктивність на одиницю ваги.

NiCd - нікель-кадмієві батареї - аналогічні NiMH, але в даний час вони виводяться з експлуатації, через токсичний кадмій і відсутність переваг над NiMH акумуляторами.

NiH2 - нікель-водневий акумулятор – екзотичні матеріали та висока ціна роблять їх використання у робототехніці не оптимальним.

Є кілька інших типів, однак у даній роботі не розглядається нічого дорогого, складного в покупці або у зарядці.

1.7.2. Порівняння Li-Ion, Li-Poly і NiMH

Фактично, батареї Li-Poly є підгрупою літій-іонних акумуляторів. Можна сказати, що вони є спеціальною версією звичайних літій-іонних акумуляторів. Чому вони особливі? Відмінність полягає в тому, що під час виробництва клітини Li-Ion повинні бути втиснуті в металеву банку (зазвичай циліндричну), щоб вони зберігали свою структурну цілісність. Li-Poly клітини, які були розроблені пізніше, мають різну конструкцію і можуть утримувати себе без підтримки зовнішнього циліндра.

Однак електричні параметри Li-Ion і Li-Poly майже ідентичні.

Акумулятори NiMH користуються популярністю через низький внутрішній опір і хороше співвідношення потужності до ваги. Вони також набагато безпечніші, ніж клітини на основі Li. Вибухи таких акумуляторів дійсно рідкісні. Питома енергія (співвідношення енергія-вага) NiMH значно гірше, ніж літієвих акумуляторів.

1.7.3. Аналіз та обґрунтування обраної системи живлення.

Мобільна платформа дослідження території має 2 колеса що приводяться в дію двигунами постійного струму. Швидкість не є критичною, оскільки фокусування об'єктивів камери, сканування кімнати за допомогою сенсорів, обчислення або передача вихідних даних займає досить тривалий час. Тому робот не може пересуватись занадто швидко.

В даному випадку на таблиці 1.3 була проаналізована потужність, а не струм, через використання багатьох перетворювачів постійного/постійного струму, які перетворюють більш високу напругу і низький струм на низьку напругу і великий струм. Тоді ми можемо вибрати мінімальну напругу для батареї - 10В - і розрахувати середній струм для нашої зручності.

Таблиця 1.3 Вимоги потужності мобільної платформи дослідження території.

Компонент	Середня потужність (W)	Максимальна потужність(W)
Мотор постійного струму	6,2	36
Мотор постійного струму	6,2	36
Камера	1	1
Міст керування двигунами	0,8	0,8
Мікрокомпютер	4	7
Wi-Fi модуль	0,7	0,9
Сумарна потужність (W):	18,9	81,7
Сумарний струм при 12V (W):	1,6	6,8

Вимоги до батарей такі:

– розмір - нижче 150 $см^3$;

– вага - нижче 500г.

– номінальна напруга - близько 10–12В.

– максимальний струм розряду - 9.3А.

– потужність - середній струм становить 2.1A, а мінімальний робочий час - 120 хвилин, тому необхідна потужність вище 4200mAh.

– низький струм саморозряду є перевагою.

NiMH: не оптимально - енергія є пріоритетом. Літієві акумулятори набагато кращі.

Li-Ion: акумуляторна батарея 6x18650 (3S2P) виготовлена з INR18650–35E Samsung 3500mAh.

– розміри: 67 x 57 x 38 мм;

– ном. напруга: 10,8 В (3,6 В на комірку);

– мін. напруга: 9В;

– макс. струм розряду: 16А;

– об'єм: 145 $см^3$;

– вага: 300г;

– місткість: 7000mAh;

– ціна: близько $ 30.

За результатами аналізу цей вибір визнаний оптимальним. Питома енергія (відношення потужності до ваги) набагато більше, ніж потрібно для предметної області. Ціна низька.

Li-Poly: акумуляторна батарея 3S1P виготовлена з LP616594 4700mAh 3.7V:

– розміри: 94 x 65 x 19 мм;

– ном. напруга: 10,8 В (3,6 В на комірку);

– мін. напруга: 9В;

– макс. струм розряду: 9.5А;

– об'єм: 113 $см^3$;

– вага: 236г;

– місткість: 4700mAh;

– ціна: близько $ 60.

Менша вага та об'єм, але гірша питома енергія та більш висока ціна. Приклад низькотемпературного типу Li-Ion - максимальний струм - лише 2C, але для предметної області цього достатньо.

У ході створення апаратної інфраструктури виявились складнощі у побудові системи живлення. Перетворювач постійного струму видавав максимальний струм у 2А, а цього виявилося недостатньо для живлення усіх компонентів. Проблема була вирішена розділом живлення моторів та іншої апаратної інфраструктури на два елементи живлення.

Для живлення двигунів використовується касета для 4 батарейок типу AA, що дає на виході загальну напругу 6В, і зображена на рисунку 1.10.

Рисунок 1.10 - Касета для 4 батарейок типу AA

Для живлення Orange Pi використовується зовнішній акумулятор 18650 Lithium Battery Charge Shield V3, зображений на рисунку 1.11. Lithium Battery Charge Shield V3 дає на виході напругу 5В і підключається до Orange Pi через вхід microUSB. Модуль автономного живлення являє собою комбінацію двох пристроїв - зарядного пристрою літієвого акумулятора і перетворювача напруги з 3В в 5В і підходить для живлення 3В і 5В пристроїв, таких як, контролери Arduino, міні-комп'ютери Orange Pi і Raspberry Pi і ін, а також зарядки мобільних пристроїв, виступаючи в якості автономного джерела живлення.

Рисунок 1.11 - 18650 Lithium Battery Charge Shield V3[16]

Для виключення вихідного напруги 5В на платі встановлений малогабаритний вимикач.

Для підключення додаткових споживачів на платі передбачені по 3 пари контактних майданчиків з напругою 3В і 5В.

1.8 Висновки

У цьому розділі було розглянуто компоненти, з який можна зібрати рухому платформу та їх основні властивості. Вибір саме цих компонентів дозволяє виконати швидку, надійну та дешеву збірку прототипу для тестування.

2 ПОБУДОВА ДІАГРАМИ РОЗГОРТАННЯ АПАРАТНОЇ ІНФРАСТРУКТУРИ

У попередньому розділі було детально розглянуто компоненти, з яких можна скласти рухому платформу. У цьому розділі буде розглянуто сам процес побудови діаграми розгортання ПАІ робота, починаючи від під'єднань джерел струму закінчуючи з'єднання мікроконтролерів з GPIO.

Цей розділ детальніше розгляне усі компоненти з точки зору їх взаємодії з навколишнім світом та іншими компонентами системи.

2.1 Під'єднання шасі до контролерів та джерела живлення

Для початку для запуску системи нам потрібно зчепити між собою усі частини шасі та підключити до нього електричне обладнання та двигуни. Шасі виготовлене з пластику і з'єднується болтами та гайками. Така система є дешевою і відкриває великі можливості для встановлення власного обладнання. Її недоліком є використання компонентів низької надійності. Колеса мають властивість змінювати кут нахилу, і міцність самої платформи, хоча достатня для дослідницької роботи, не є оптимальною для промислового використання.

У передній частині конструкції розміщена відеокамера, що буде відповідати за отримання зображення. У задній - два двигуни, що під'єднані до коліс - це буде слугувати приводом платформи. Ці частини надійно з'єднані болтами. У середині системи розмістимо корпус Orange Pi - через нього буде з'єднано джерело струму з двигунам, а також плата H-Bridge L9110, що буде регулювати подачу струму на двигуни. Саме джерело струму буде розміщено зверху корпусу мікрокомп'ютера. В задній частині системи розміщений акумулятор живлення 18650 Lithium Battery Charge Shield V3.

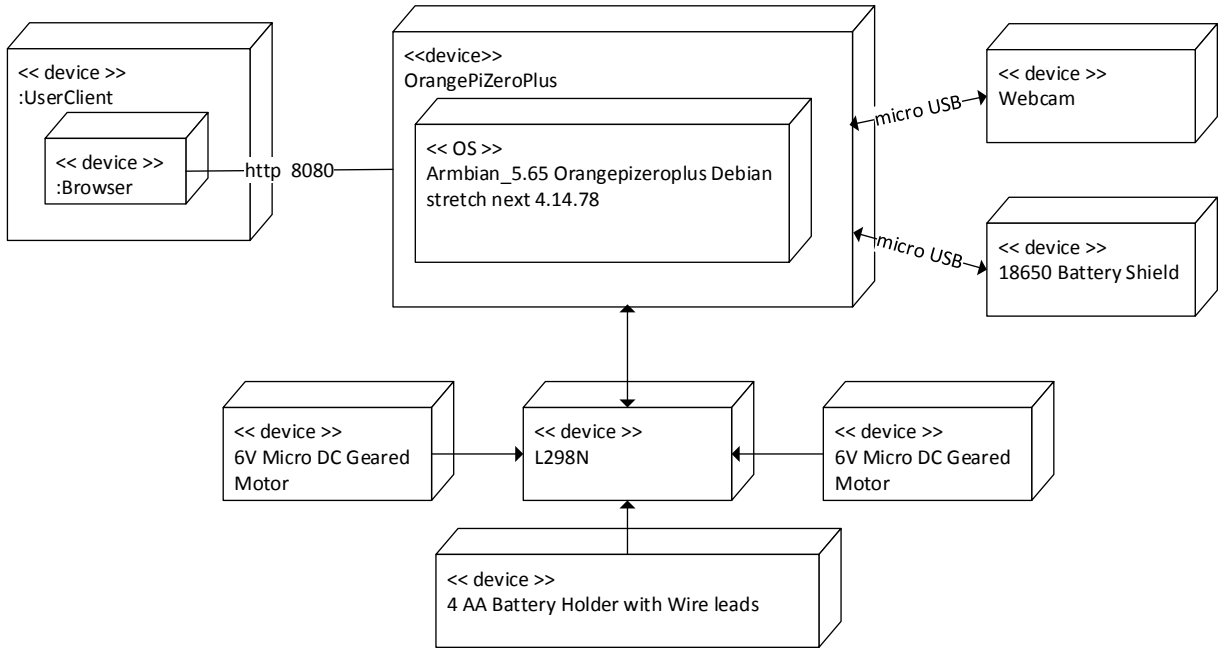

Рисунок 2.2 – Діаграма розгортання апаратної інфраструктури

На рис 2.2 показана діаграма розгортання наземного робота. В нижній частині діаграми розгортання зображені двигуни і їх система живлення. Оскільки робота двигунів викликає різкі зміни напруги, живлення двигунів і мікрокомп'ютера відбувається через окремі ланцюги. Таким чином, двигуни керуються через інтерфейс General Purpose Input-Output (GPIO) і плату H-Bridge L9110. У верхній частині рисунку показана плата мікрокомп'ютера Orange Pi, і операційна система Armbian, встановлена на ній. До мікрокомп'ютера за допомогою USB під'єднується живлення та веб-камера. WI-FI, встановлений у плату мікрокомп'ютера, дозволяє підключитися до встановленого-веб-сервера для керування наземним роботом. Кінцевий вигляд створеного прототипу НРДТ зображений на рисунку 2.3.

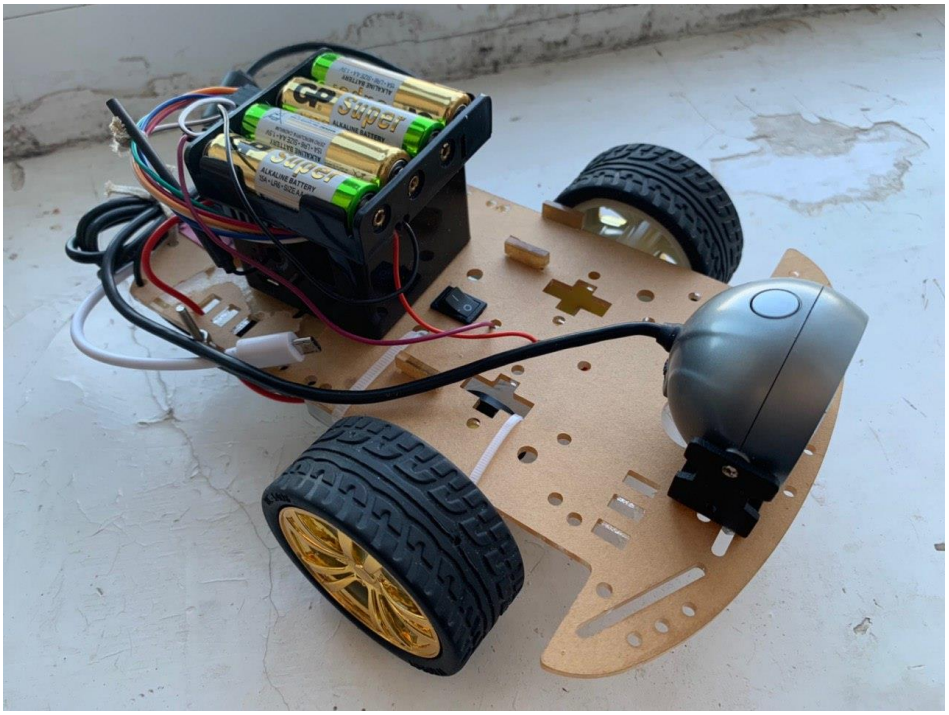

Рисунок 2.3 - Кінцевий вигляд мобільної платформи

2.2 Налаштування операційної системи

Orange Pi Zero Plus поставляється без операційної системи, її потрібно записати на картку пам’яті і встановити. Для проведення цієї операції, окрім самої плати, потрібно підготувати наступні пристрої[6]:

– MicroCD картка, мінімум 8 Гб. Клас 10. Бажано використовувати офіційні картки, оскільки надійність є дуже важливою;

– Ethernet кабель. Плата не має зручного відео виходу, тому налаштування, у тому числі підключення до Wi-Fi, виконується через протокол SSH;

– адаптер живлення на 5В та 2А. Дуже важливо щоб адаптер дійсно міг видавати струм у 2А, мікрокомп’ютер є досить вибагливим з точки зору енергоспоживання.

Для нормального використання Orange Pi необхідно спочатку встановити операційну систему в TF-карту.

Вставляючи картку TF у комп'ютер, ємність картки повинна бути більше, ніж операційна система, зазвичай необхідно 8 Гб або більше. Спочатку TF картку необхідно відформатовати. Для цього необхідно виконати наступні дії:

– завантажити інструменти для форматування картки TF, такі як TF Formatter, який можна завантажити з офіційного сайту[19];

– розпакувати завантажені файли та запустіть файл setup.exe;

– у налаштуваннях параметрів вибрати кнопку "format" для швидкого вибору форматування. У графі "Налаштування розміру формату" вибрати "(ON)";

– переконатися, що вставлений диск TF картки відповідає вимогам;

– натиснути кнопку "Формат";

– завантажити файл образу операційної системи зі сторінки завантаження сайту плати мікрокомпютера[20];

– завантажити інструменти для запису зображення, наприклад Win32 Diskimager, зі сторінки завантаження[21];

– вибрати шлях розпакування файлу зображення;

– натиснути кнопку "Записати" і почекати, поки образ буде записано;

– після закінчення запису образу натиснути кнопку "Вихід".

2.3 Налаштування серверу

Програмна інфраструктура являє собою клієнт-серверне ПО для керування мобільною платформою. Воно представляє собою сервер для отримання команд та клієнт у вигляді браузерного JavaScript додатку, яке реагує на натискання клавіш “вперед”, “назад”, “вліво” та “вправо” та відправляє ці команди серверу на виконання. Сервер знаходиться на Orange Pi, клієнт динамічно завантажується у браузер із сервера. Тобто для керування має бути налаштований HTTP-зв’язок у локальній або глобальній мережі. Архітектура даного ПО буде в деталях описана нижче, в цьому розділі розглянемо процес встановлення програмної інфраструктури на плату мікроконтролера.

Бібліотека WiringPi завантажена на GIT для полегшення відстеження змін. Для того, що б витягнути WiringOP з github необхідно щоб був встановлений

пакет git-core. Якщо не встановлений GIT, то можна встановити його за допомогою команди:

sudo apt-get install git-core

При виникненні будь-яких помилок, тоді необхідно оновити ОС наступними командами:

sudo apt-get update

sudo apt-get upgrade

Далі необхідно завантажити WiringOP/WiringPi. Необхідно зауважити, що основна програма несумісна з Orange Pi Zero Plus, тому у ході виконання роботи були виконані доповнення для забезпечення сумісності:

git clone https://github.com/roman-bessmertnyi/WiringOP-Zero

У разі якщо файли вже були клоновані раніше, тоді необхідно просто оновити до останньої версії:

cd WiringOP

git pull origin

Установка WiringOP/WiringPi (H3, H5):

cd WiringOP

chmod +x ./build

sudo ./build

Перевірка WiringOP/WiringPi

gpio -v

gpio readall

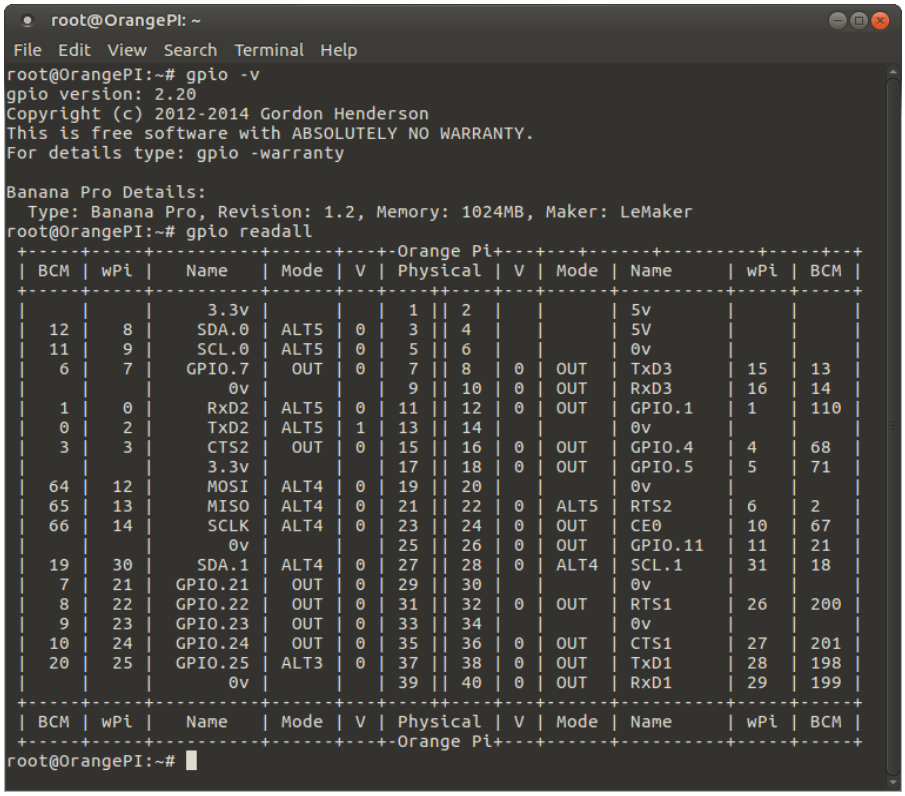

Рисунок 2.4 – Результат перевірки роботи WiringOP/WiringPi

Програмна архітектура мобільної платформи працює використовуючи технологію Java, тому необхідно встановити JDK:

sudo apt-get install default-jdk

java –version

Для забезпечення контролю версій ПО розміщене на платформі github. Відповідно для роботи необхідно завантажити вихідний код і побудувати програмну інфраструктуру. Для завдання побудови артефакту використовується Maven, для його встановлення використовуються наступні команди:

sudo apt install maven

mvn -version

Одним з базових елементів захоплення зображення є бібліотека Video4Linux4Java (v4l4j):

git clone https://github.com/roman-bessmertnyi/v4l4j

cd v4l4j

mvn install:install-file -Dfile= v4l4j.jar -DgroupId=brs -DartifactId=v4l4j -Dpackaging=jar -Dversion=0.9.1 -DgeneratePom=true

Для встановлення програмного забезпечення відповідно необхідно виконати наступні команди:

cd

git clone https://github.com/roman-bessmertnyi/donkeybug

cd donkeybug

mvn clean validate compile test package

Наостанок наведемо команди для запуску і зупинки сервера:

sudo java -jar target/donkeybug-1.0.jar --OS="Linux"

sudo pkill -9 -f target/donkeybug-1.0.jar

2.4 Висновки

У цьому розділі було розглянути процес побудови рухомої платформи прототипу для тестування, клієнта для керування та перегляду потокового відео з рухомої платформи розраховано затримки на виконання команд.

Для перевірки системи на коректність роботи було проведено інтеграційне тестування. Всі частини системи були з’єднані включно з джерелами живлення. На стороні Orange Pi було запущено сервер, що приймає запити на рух системи та передає зображення з камери на екран клієнту. Результати тестування показали, що все налаштовано та функціонує коректно та очікувано.

3 ПОБУДОВА ДІАГРАМИ КОМПОНЕНТІВ ПРОГРАМНО-АПАРАТНОЇ ІНФРАСТРУКТУРИ РОБОТА

3.1 Діаграма компонентів

Для більш докладного опису результатів дослідження пропонується UML діаграма компонентів, що підтверджена результатами дослідно-конструкторської роботи і показана на рис.3.1. На даному рисунку виділені ті компоненти мобільної платформи, що відображають базові елементи систем передачі команд керування і передачі поточного відео с рухомої системи.

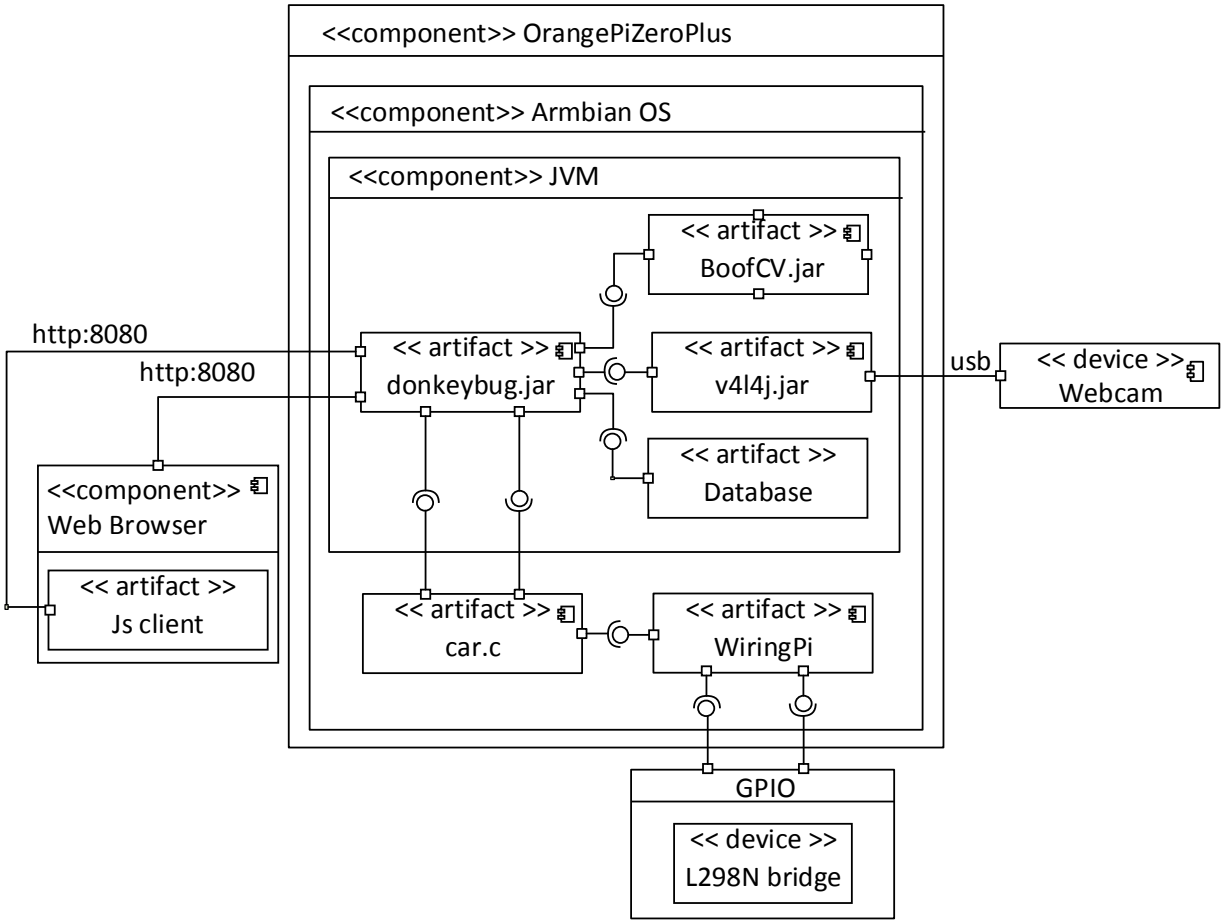

Рисунок 3.1 – Діаграма компонентів

Розглянемо діаграму компонентів більш докладно. На ній виділені базові елементи систем передачі команд керування і передачі поточного відео с рухомої системи. Зображення з відеокамери, що використовується для орієнтації у просторі, передається через інтерфейс USB. Зображення захоплюється з використанням бібліотеки Video4Linux4Java, що буде описана докладно далі.

Інформація зображення може бути накопичена у базі даних, оброблена нейронною мережею для автономної навігації або надана оператору через веб інтерфейс для додаткового контролю.

3.2 Мова програмування Java

Java — об'єктно-орієнтована мова програмування. Програми на Java транслюються в байт-код, що виконується віртуальною машиною Java (JVM) — програмою, що перероблює байтовий код і передає інструкції обладнанню як інтерпретатор.

Перевагою подібного способу виконання програм є повна незалежність байт-коду від операційної системи і обладнання, що дозволяє виконувати Java-додатки на будь-якому пристрої, для якого існує відповідна віртуальна машина.

Іншою важливою особливістю технології Java є гнучка система безпеки, в рамках якої виконання програми повністю контролюється віртуальною машиною. Будь-які операції, які перевищують встановлені повноваження програми (наприклад, спроба несанкціонованого доступу до даних або з'єднання з іншим комп'ютером), викликають негайне переривання.

3.3 Spring Framework

В якості базової технології в дослідній роботі був використаний фреймворк Spring, що працює на Java Virtual Machine. Spring Framework[22] — це програмний каркас з відкритим кодом та контейнери з підтримкою інверсії управління для платформи Java. Основні особливості Spring Framework можуть бути використані будь-яким додатком Java, але для даної роботи актуальним є розширення для створення веб-додатків.

Spring Framework поділений на модулі. Програми можуть вибрати, які модул і вони потребують. В центрі знаходяться модулі контейнера ядра, включаючи модель конфігурації і механізм ін'єкції залежностей. Крім того, Spring Framework забезпечує фундаментальну підтримку різних архітектур прикладних програм, включаючи обмін повідомленнями, транзакційні дані,

персистентність і веб. Вона також включає в себе Веб-сервіс на основі веб-платформи Servlet і, паралельно, реактивний веб-фреймворк Spring WebFlux.

3.4 Веб-сервер ApacheTomcat

Веб-сервер — сервер, що приймає HTTP-запити від клієнтів веб-браузерів і видає їм HTTP-відповіді разом з HTML-сторінкою, зображенням, файлом, медіа-потоком або іншими даними. Веб-сервером називають як програмне забезпечення, яке виконує функції веб-сервера, так і безпосередньо комп'ютер, на якому це програмне забезпечення працює.

Клієнт, яким зазвичай є веб-браузер, передає веб-серверу запити на отримання ресурсів, позначених URL-адресами. Ресурси — це HTML-сторінки, зображення, файли, медіа-потоки або інші дані, які необхідні клієнту. У відповідь веб-сервер передає клієнту запитані дані. Цей обмін відбувається по протоколу HTTP.

Сервлет є інтерфейсом Java, реалізація якого розширює функціональні можливості сервера. Сервлет взаємодіє з клієнтами за допомогою принципу запит-відповідь. Хоча сервлети можуть обслуговувати будь-які запити, вони зазвичай використовуються для розширення веб-серверів. Для таких додатків технологія Java Servlet визначає HTTP-специфічні сервлет класи.

Контейнер сервлетів — програма, що представляє собою сервер, який займається системною підтримкою сервлетів і забезпечує їх життєвий цикл відповідно до правил, визначеними в специфікаціях. Може працювати як повноцінний самостійний веб-сервер, бути постачальником сторінок для іншого веб-сервера, наприклад Apache, або інтегруватися в Java EE сервер. Забезпечує обмін даними між сервлетом і клієнтами, бере на себе виконання таких функцій, як створення програмного середовища для функціонуючого сервлету, ідентифікацію та авторизацію клієнтів, організацію сесії для кожного з них.

Tomcat також надає повну цілісність даних і коду, виділяючи бізнес логіку на окремий сервер, або на невелику кількість серверів, можна гарантувати

оновлення та покращення додатків для всіх користувачів. Відсутній ризик, що стара версія програми отримає доступ до даних або зможе їх змінити старим несумісним образом.

Централізоване налаштування і управління вносять зміни в налаштування програми, такі як зміна сервера бази даних або установок системи, які можуть проводитися централізовано.

Також існує безпека серверних додатків яка діє як центральна точка, використовуючи яку, постачальники сервісів можуть керувати доступом до даних і частин самих додатків, що вважається перевагою захисту. Її наявність дозволяє перемістити відповідальність за автентифікацію з потенційно небезпечного рівня клієнта на рівень сервера додатків, при цьому додатково приховуючи рівень бази даних.

Підтримка транзакцій також є необхідною. Транзакція є одиницю активності, під час якої велика кількість змін ресурсів може бути виконана атомарно. Кінцеві користувачі при цьому можуть виграти від стандартизованої поведінки системи до зменшення часу на розробку і зниження вартості. У той час як сервер додатків виконує масу потрібного генерування коду, розробники можуть сфокусуватися над бізнес-логікою.

Apache Tomcat дозволяє запускати веб-додатки, містить ряд програм для самоконфігурації. Tomcat використовується в якості самостійного веб-сервера, як сервер контенту в поєднанні з веб-сервером Apache HTTP Server, а також в якості контейнера сервлетів в серверах додатків JBoss і GlassFish.

3.5 Video4Linux4Java

Одним з базових елементів захоплення зображення є бібліотека Video4Linux4Java (v4l4j)[23], що може бути інтегрована у Spring Framework. Вона реалізуєтся як Java-пакет, що забезпечує простий доступ до інтерфейсу захоплення відео в API Video4Linux (V4L). У результаті дослідно-конструкторської роботи було виявлено, що саме v4l4j надає найбільш

стабільний і швидкий доступ до відео інформації с веб-камери. Webcam Capture надає дані у форматі BufferedImage, що оптимізований для швидкої трансформації у інші формати. Для збереження, аналізу і виводу зображення конвертується у масив байтів у форматі jpeg. Бібліотека v4l4j дає розробнику можливості розробляти програми, що дозволяють наступне:

– отримувати інформацію про відеопристрій, наприклад, кількість і тип відеовходів, підтримувані формати зображень, роздільну здатність і стандарти відео та доступні тюнери;

– кадри захоплення з будь-яких пристроїв із підтримкою V4L, включаючи USB / Firewire Веб-камери, карти захоплення відео та карти тюнерів;

– доступ до елементів керування відео, таких як яскравість, контрастність, коефіцієнт посилення, нахил, фокус;

– тюнери доступу та керування (частота отримання / встановлення, доступ до потужності прийнятого сигналу.

3.6 WiringPi

Для управління апаратними виводами GPIO була обрана бібліотека WiringPi[24]. Ця бібліотека дає доступ до плати контролера. Вона, написана на мові C для пристроїв BCM2835, BCM2836 і BCM2837 SoC, що використовуються в основному в Raspberry Pi, але портована для багатьох платформ, у тому числі OrangePi Zero Plus. Випускається під ліцензією GNU LGPLv3 і може використовуватися з багатьма мовами програмування з відповідними обгортками

3.7 Sqlite

Для подальшого використання відео дані зберігаються. У ході дослідно-конструкторської роботи зображення зберігалися локально, проте використання хмарних файлосховищ також можливе. Посилання на зображення, разом з метаданими зберігаються у локальній Sqlite[25] реляційній базі даних.

3.8 Бібліотека Hibernate

У розробці програмної інфраструктури для зберігання даних, поміж іншого, була використана бібліотека Hibernate[26]. Hibernate це засіб об'єктно-реляційного відображення(object-relational mapping, ORM)[27] для платформи Java. Hibernate є вільним програмним забезпеченням і надає легкий для використання каркас (фреймворк) для відображення між об'єктно-орієнтованою моделлю даних і традиційною реляційною базою даних.

Hibernate — бібліотека для мови програмування Java, призначена для вирішення завдань об'єктно-реляційного відображення (objectrelational mapping — ORM).

Метою Hibernate є звільнення розробника від значного обсягу низькорівневого програмування щодо забезпечення зберігання об'єктів в реляційній базі даних. Розробник може використовувати Hibernate як в процесі проектування системи класів і таблиць «з нуля», так і для роботи з вже існуючою базою даних.

Hibernate не тільки вирішує завдання зв'язку класів Java з таблицями бази даних (і типів даних Java з типами даних SQL), а й також надає засоби для автоматичної генерації і оновлення набору таблиць, побудови запитів і обробки отриманих даних, а також може значно зменшити час розробки, яке зазвичай витрачається на ручне написання SQL і JDBC-коду. Hibernate автоматизує генерацію SQL-запитів і звільняє розробника від ручної обробки результуючого набору даних і перетворення об'єктів, максимально полегшуючи перенесення додатку на будь-які бази даних SQL.

Hibernate забезпечує прозору підтримку збереження даних (persistence) для «POJO» (тобто для стандартних Java-об'єктів); єдина сувора вимога для зберігання класу — наявність конструктора за замовчуванням (без параметрів). Для коректної поведінки в деяких додатках потрібно також приділити увагу методам equals () і hashCode ().

3.9 Система управління базами даних

В додатку, що розробляється необхідно постійно впорядковано зберігати набір даних, з якими працює користувач. Для цієї мети найраціональнішим рішенням буде використання бази даних. Для управління створення бази даних буде використовуватися система управління базами даних.

SqlLite - це вільно поширювана об'єктно-реляційна система управління базами даних (ORDBMS), найбільш розвинена з відкритих СУБД в світі і є реальною альтернативою комерційних баз даних.

Клієнтський запит проходить наступні стадії.

– підключення до бази даних;

– парсинг, де перевіряється коректність запиту і створюється дерево запиту (query tree), в основу парсера покладені базові юніксові утиліти yacc і lex;

– перезапис: береться дерево запитів і перевіряється наявність в ньому правил (rules), які лежать в системних каталогах; всякий раз користувальницький запит переписується на запит, який отримує доступ до таблиць бази даних;

– оптимізатор: на кожен запит створюється план запиту - query plan, який передається виконавцю – executor; сенс плану в тому, що в ньому перебираються всі можливі варіанти отримання результату (чи використовувати індекси, джойни і т.д.), і вибирається найшвидший варіант;

– виконання запиту: виконавець рекурсивно проходить по дереву і отримує результат, використовуючи при цьому сортування, джойни і т.д., і повертає рядки. SqlLite - об'єктно-реляційна база даних, кожна таблиця в ній представляє клас, між таблицями реалізовано успадкування. Реалізовано стандарти SQL92 і SQL99.

– транзакційна модель побудована на основі так званого multi-version concurrency control (MVCC), що дає максимальну продуктивність; посилальна цілісність забезпечена наявністю первинних і вторинних ключів.

3.10 Tensorflow

Для автоматизації контрою якості зображення доцільним було визначено використання нейронної мережі-класифікатора. Для розробки даної мережі пропонується використовувати платформу Tensorflow[28] у зв'язку з Tensorflow detection model zoo[29].

TensorFlow - це відкрита платформа з відкритим вихідним кодом для машинного навчання. Вона має всеосяжну, гнучку екосистему інструментів і бібліотек, які дозволяють швидко створювати і розгортати додатки машинного навчання. TensorFlow була портована для багатьох мов програмування, включаючи Java. Tensorflow detection model zoo дозволяє використовувати претреновані нейронні мережі для задач класифікації.

3.11 Висновки

Були визначені основні архітектурні особливості та складові системи. Був проведений аналіз вимог та предметної області, та створені специфікації роботи програми.

4 РЕАЛІЗАІЯ ПРОГРАМНОЇ ЧАСТИНИ ІНФРАСТРУКТУРИ РОБОТА

4.1 Сценарії використання системи

На рисунку 4.1 зображено Use Case діаграма наземної платформи дослідження території.

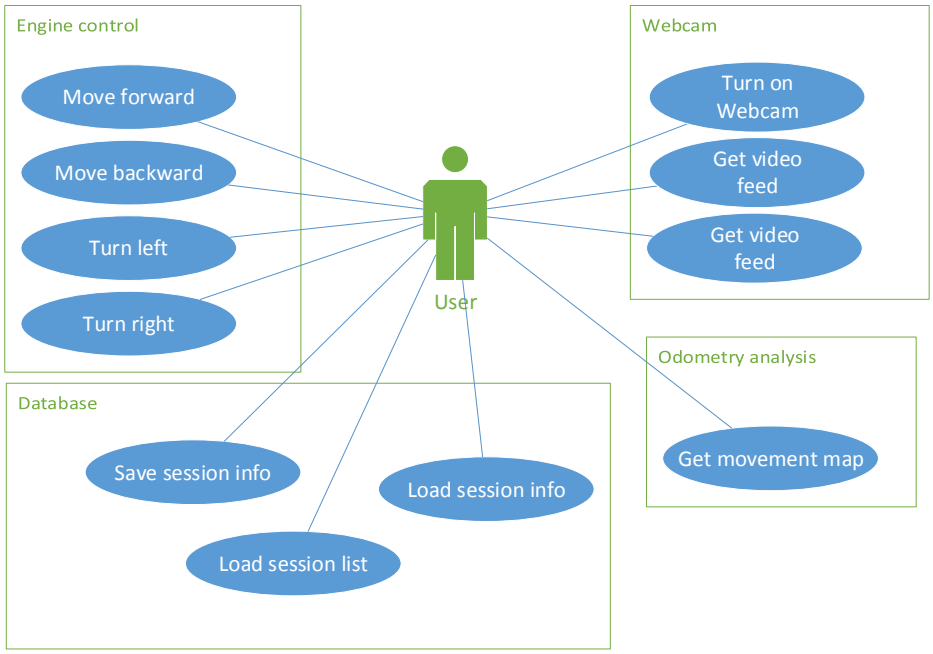

Рисунок 4.1 – Use Case діаграма наземної платформи

Опишемо наведену діаграму більш детально. Першою основною функціональною категорією є управління двигунами, тобто система прийому, обробки, передачі та виконання команд керування.

Структура апаратної інфраструктури диктує обсяг команд керування. Мобільна платформа може рухатись вперед, назад, повертати вліво або вправо, або ж стояти на місті. Для забезпечення контрою мобільної платформи необхідно, щоб команди передавались швидко. Особливо важливим є зупинка мобільної платформи вразі відсутності команди, або втрати зв'язку.

Використання клієнт-серверної архітектури докладає додаткові умови. Клієнт відповідає за наявність зручного та зрозумілого інтерфейсу, за швидку передачу команд, а також за наявність зв'язку та підтвердження виконання команд. Відповідно сервер відповідає за отримання та обробку команд. Враховуючи, що сервер, окрім клієнта, обмінюється даними з програмою

керування двигунами, сервер відповідає за наявність зв’язку та адекватну передачу команд в обох випадках.

Для типової команди керування двигунами побудуємо таблиці із наступними полями: назва діяльності, її унікальний ідентифікатор, опис, актор, тригер спрацьовування, попередні умови для виконання, умови після виконання, основний та альтернативний потоки розвитку дій та можливі помилки.

Розглянемо діяльність «Move forward». Її детальний опис наведено у таблиці 4.1.

Таблиця 4.1 – Опис діяльності « Move forward »

Найменування	Move forward
ID	C_01
Опис	Після початку сесії керування та підтвердження зв’язку з сервером, користувач може подати команду на початок руху вперед. Користувач натискає на відповідну стрілку. Стрілка візуально підкреслюється для підтвердження натискання.
Актори	Користувач системи
Тригери	Користувач натискає кнопку руху вперед.
Передумови	Користувач розпочав сесію керування та отримав підтвердження дієздатності мобільної платформи
Післяумови	Клієнт продовжує посилати команду поки натиснута кнопка руху. Сервер вмикає двигуни і утримує їх у стані руху поки від клієнта продовжують приходити команди.
Основний потік розвитку	– Користувач натискає на стрілку руху вперед. – Клієнт змінює вигляд кнопки. – Клієнт відсилає запит на сервер. – Сервер визначає отриману команду. – Сервер посилає команду до програми керування двигунами.

	– Програма керування двигунами вмикає двигун. – Сервер очікує отримання наступної команди.
Можливі помилки	– Відсутність звязку. Користувач буде повідомнений про втрату звязку, клієнт спробує відновити зв'язок. – Відсутність звязку з програмою керування двигунами. Сервер перестає посилати команди, відсилає повідомлення користувачу і намагається перезапустити програму керування двигунами.

Команда зупинки руху виявляється найбільш важливою, оскільки продовження руху без команди від системи управління може викликати розрядку і виснаження джерела струму, а також зіткнення платформи з перешкодами та падіння платформи У випадку, якщо сервер не отримує підтвердження продовження руху через визначений час, до програмы керування надходить команда зупинки. За нормальної роботи клієнт посилає команду зупинки коли користувач перестає утримувати кнопку руху. Далі клієнт продовжує посилати команди зупинки для підтвердження наявності зв’язку і підтвердження зупинку у випадку якщо перша команда не була успішно оброблена.

Другою функціональною категорією є передача поточного відео с рухомої системи. Отримання даних починається з клієнта. Спочатку проходить валідація запиту. Клієнт перевіряє наявність зв’язку. Якщо сервер не вкладається у відведений час на видачу кадру, запит відміняється щоб уникнути зайвого навантаження.

На сервері перш ніж видати зображення проводиться підключення камери та відповідних драйверів, що можуть відрізнятися залежно від операційної системи та конфігурації програмного забезпечення.

Після отримання запиту, сервер перевіряє дієздатність камери, після чого дає запит на отримання зображення. Після виконання запиту, отримане зображення перетворюється у бінарний формат і відправляється на клієнт. У

випадку помилки в отриманні зображення клієнт отримує повідомлення про несправність камери.

На стороні клієнта отримане зображення зберігається у пам’яті браузера і відображається за допомогою посилання на збережене зображення.

Була розглятута діяльність «Turn on webcam». Її детальний опис наведено у таблиці 4.2.

Таблиця 4.2 – Опис діяльності «Turn on webcam»

Назва	Turn on webcam
ID	C_02
Опис	Після успішного запуску сервера обирається бібліотака та параметри захоплення зображення залежно від операційної системи та конфігурації. Ці дані інкапсулюються в відповідному об’єкті, що надає єдиний інтерфейс доступу до відеокамери.
Актори	Користувач системи
Тригери	Користувач вмикає сервер.
Передумови	Сервер успішно запустився і підтвердив свою дієздатність.
Післяумови	Обєкт відеокамери готовий до використання для передачі потокового відео на клієнт.
Основний потік розвитку	– Користувач вмикає сервер. – Сервер підключає компоненти роботи з відеокамерою. – Сервер оброблює логіку налаштування відеокамери. – Сервер запамятовує параметри для подальшої роботи з відеокамерою.
Можливі помилки	– Обраний драйвер може бути несумісним з встановленою операційною системою. – Обрані налаштування можуть бути несумісними з обраною відеокамерою.

	– Відсутність звязку з відеокамерою.

Була розглятута діяльність «Get video feed». Її детальний опис наведено у таблиці 4.3.

Назва	Get video feed
ID	C_03
Опис	Після отримання запиту, сервер перевіряє дієздатність камери, після чого дає запит на отримання зображення. Після виконання запиту, отримане зображення перетворюється у бінарний формат і відправляється на клієнт. На стороні клієнта отримане зображення відображається на екрані.
Актори	Користувач системи
Тригери	Користувач розпочинає сесію керування.
Передумови	Успішне підключення відеокамери.
Післяумови	Клієнт бачить відеоданні на екрані.
Основний потік розвитку	– Користувач розпочинає сесію керування. – Сервер дає запит на отримання зображення. – Сервер оброблює зображення у відповідний формат. – Сервер надсилає оброблені дані. – Клієнт виводить зображення на екран.
Можливі помилки	– Обраний драйвер може бути несумісним з встановленою операційною системою. – Обрані налаштування можуть бути несумісними з обраною відеокамерою. – Відсутність звязку з відеокамерою.

Була розглятута діяльність «Get video feed». Її детальний опис наведено у таблиці 4.3.

Назва	Get FPS

ID	C_03
Опис	Після отримання запиту, сервер перевіряє дієздатність камери, після чого дає запит на отримання кільості кадрів в секунду. Після виконання запиту, отримані дані відправляється на клієнт. На стороні клієнта отримані дані відображається на екрані.
Актори	Користувач системи
Тригери	Користувач розпочинає сесію керування.
Передумови	Успішне підключення відеокамери.
Післяумови	Клієнт бачить кількість кадрів в секунду на екрані.
Основний потік розвитку	– Користувач розпочинає сесію керування. – Сервер дає запит на отримання даних. – Сервер надсилає оброблені дані. – Клієнт виводить кількість кадрів в секунду на екран.
Можливі помилки	– Обраний драйвер може бути несумісним з встановленою операційною системою. – Обрані налаштування можуть бути несумісними з обраною відеокамерою. – Відсутність звязку з відеокамерою.

Інші активності мають схожі властивості та можливі випадки поведінки, тому їх детальний огляд буде опущено.

4.2 Система передачі команд керування

Для більш докладного опису роботи системи пропонується UML діаграма послідовності передачі команд керування.

При розробці мобільної платформи була виявлена важливість забезпечення зупинки. Така необхідність може виникнути в разі відсутності сигналу, помилки в роботі однієї з систем або відсутності команди користувача. Реалізація цієї вимоги відбувається за допомогою використання кінцевих автоматів на всіх рівнях архітектури – клієнті та сервері.. За виконання необхідних умов стан кінцевого автомата повертається у стан спокою, зупиняючи платформу.

Під час керування мобільною платформою, клієнт app.js(рис 4.2) працює в циклі. У ньому клієнт посилає стан кінцевого автомата клієнта на сервер і чекає на підтвердження виконання команди. Сервер працює за архітектурою MVC, тому логіка виконання команди поділена на класи CommandController, PiCarService, CommandSender та CarAppManager. Далі їх робота розглянута детальніше.

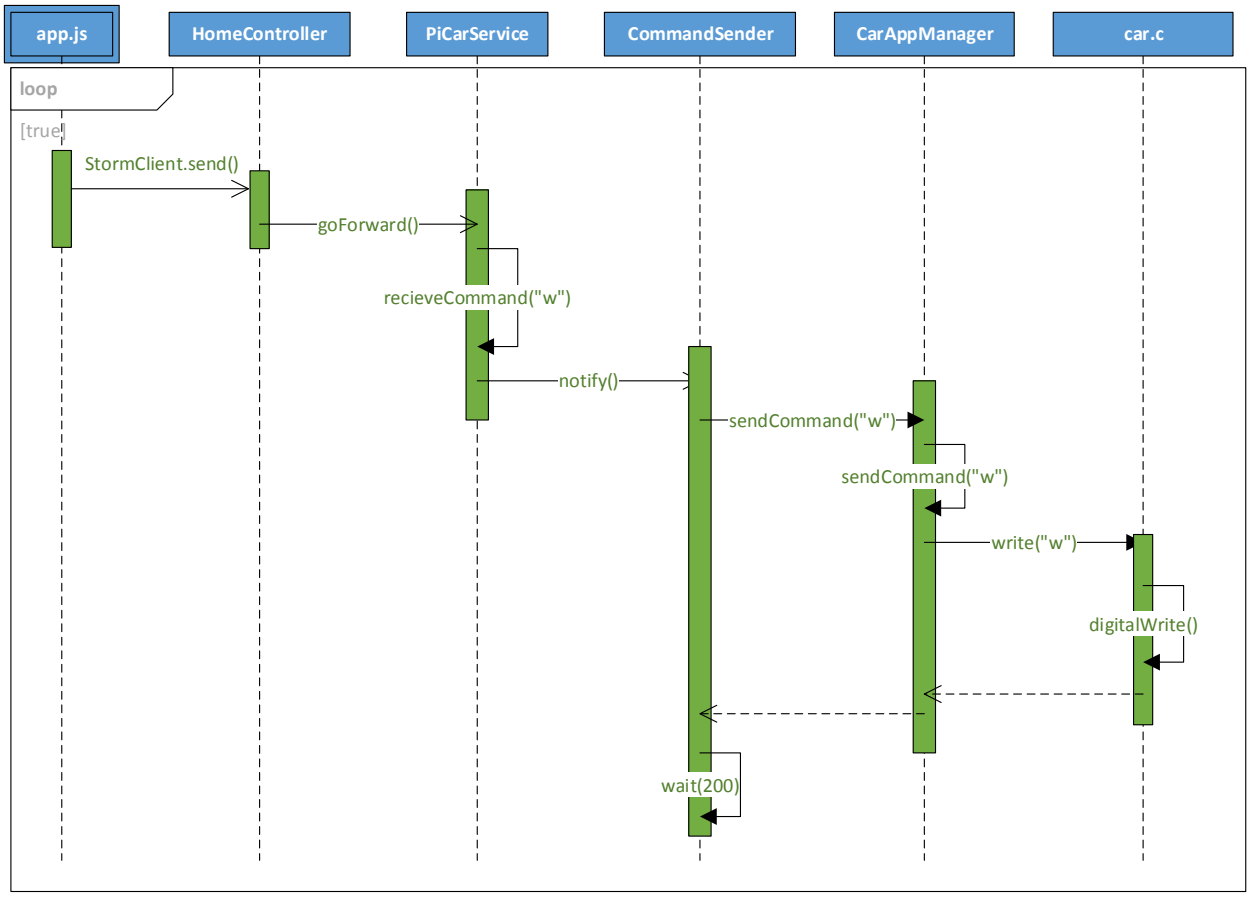

Рисунок 4.2 – Діаграма послідовності передачі команд керування.

CommandController інкапсулює обробку запиту, викликаючи відповідний метод сервісу ContinuousCarService для подальшої його обробки. Коли ви робите

запит до програми, контролер відповідає за повернення відповіді на цей запит. Контролер може виконувати одну або більше дій. Дія контролера може повертати різні типи результатів дії до певного запиту. Контролер отримує вхід від користувачів через перегляд, потім обробляє дані користувача за допомогою моделі і передає результати назад у перегляд. У разі отримання неправильної команди клас посилає команду зупинки.

Для обробки відповідної команди використовується клас ContinuousCarService. Він представляє собою серверний кінцевий автомат, і відповідає за трансформацію команд з клієнта у команди до програми керування двигунами car.c(рис 4.2). Сервер відповідає за зупинку платформи у разі відсутності зв’язку. Для підтвердження наявності зв’язку команди з клієнта приходять у циклі через зазначені інтервали у 100 мс, для забезпечення. У той же час команди до програми керування двигунами(далі ПКД) поступають лише в разі зміни стану автомату.

Оскільки обробка стану серверного автомату має виконуватись паралельно, використовується допоміжний клас CommandSender. Клас ContinuousCarService тоді відповідає за створення обєкту класу CommandSender та передачі йому отриманих команд. Після отримання команди клас CommandSender чекає отримання наступної. Отримана команда звіряється з попередньою, і якщо від клієнта поступила нова команда, вона передається ПКД. У випадку якщо команда не поступила протягом 200 мс, вважається що зв'язок не є стабільним для забезпечення керування або втрачений, і подається команда зупинки.

Програма керування двигунами написана мовою програмування С, тому виникла необхідність обміну даними з сервером. Архітектурно ПКД виконана як консольна програма, що приймає команди і надсилає результат виконання. Клас CarAppManager інкапсулює роботу з консольним додатком, і надає інтерфейс надсилання команд, що використовується ContinuousCarService. ПДК запускається як дочірній процес, і клас CarAppManager перехоплює потоки

входу-виходу, програмно надсилаючи консольні команди. Надіславши команду, клас очікує відповіді від ПДК, і повертає цю відповідь як результат своєї роботи.

4.3 Система передачі поточного відео

Розглянемо роботу системи за допомогою UML діаграми послідовності передачі поточного відео.

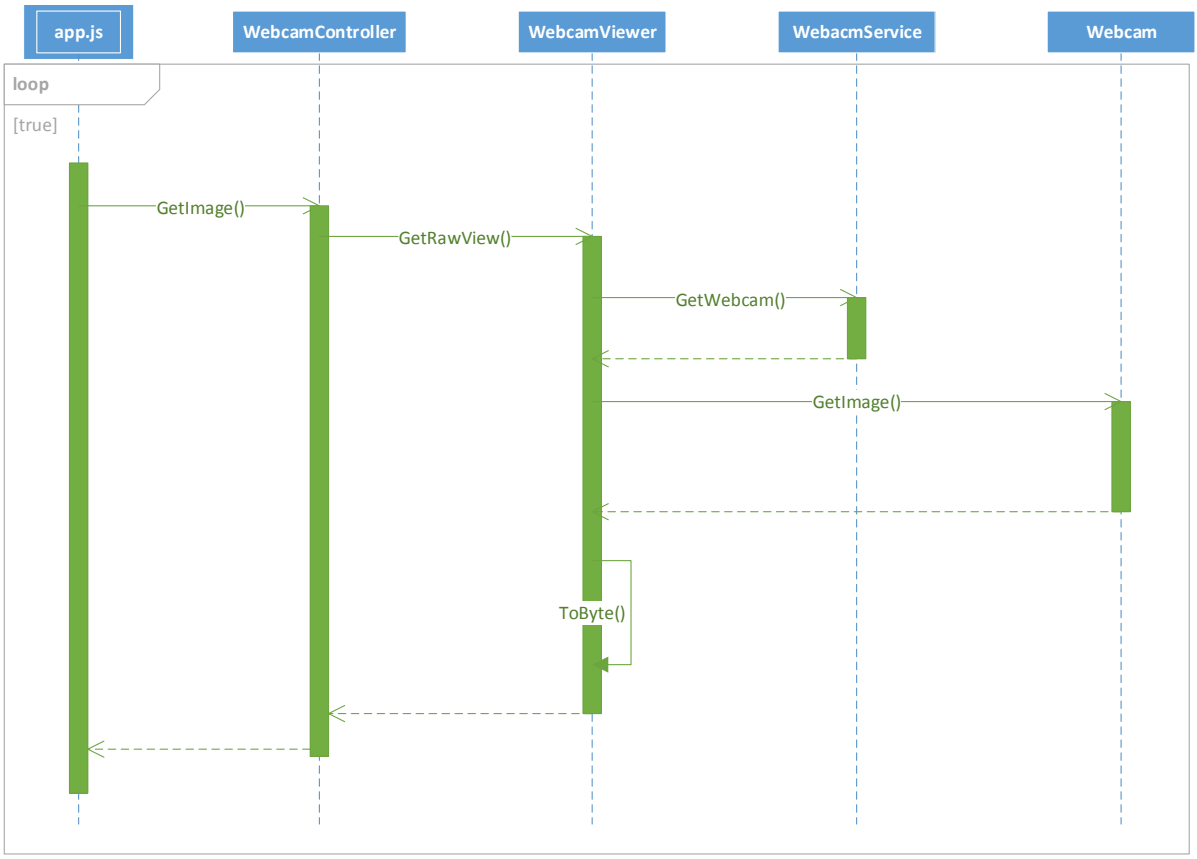

Рисунок 4.3 – Діаграма послідовності передачі поточного відео.

Під час керування мобільною платформою, клієнт app.js(рис 4.3) працює в циклі. Цикл викликається тридцять разів на секунду, для утворення відео потоку. Клієнт посилає запит на отримання зображення на сервер. Сервер працює за архітектурою MVC, тому логіка отримання зображення поділена на класи WebcamController, WebcamViewer та WebcamService. Розглянемо їх роботу детальніше.

WebcamController інкапсулює обробку запиту, викликаючи відповідний метод сервісу WebcamViewer для подальшої його обробки. Коли ви робите запит до програми, контролер відповідає за повернення відповіді на цей запит.

Контролер може виконувати одну або більше дій. Дія контролера може повертати різні типи результатів дії до певного запиту. Контролер отримує вхід від користувачів через перегляд, потім обробляє дані користувача за допомогою моделі і передає результати назад у перегляд. У разі отримання неправильної команди клас посилає команду зупинки.

Налаштування роботи камери виконується класом WebcamService, що зчитує конфігурацію сервера і повертає об’єкт Webcam відповідно до зазначених специфікацій. До налаштувань входять розширення відео та драйвер, використаний для роботи з камерою.

Для обробки запиту використовується клас WebcamViewer. Він інкапсулює процес отримання зображення у сервері. Для роботу з відеокамерою викликається об’єкт класу Webcam, до якого виконується запит отримання зображення. За відсутності зображення повертається програмно утворений чорний квадрат, і повідомлення до клієнта про відсутність зв’язку з відеокамерою. Отримане зображення конвертується у масив байтів за допомогою стандартного класу ByteArrayOutputStream.

4.4 Система збереження даних дослідження території

Для дослідження території бажаним є збереження отриманої інформації. Це дозволяє порівнювати дані дослідження території та використовувати методи математичного аналізу та моделювання для отримання висновків щодо стану території. Наприклад, систематичне збереження даних стану будівлі після пожежі дозволяє розробити ефективніший план відбудови приміщень.

Приймаючи до уваги актуальність системи збереження інформації для мобільної платформи дослідження території була теоретично розроблена і обґрунтована архітектура роботи підсистеми збереження даних. Розглянемо роботу системи за допомогою діаграми потоку даних, зображеної на рисунку 4.4.

Система зберігає три сутності даних – зображення, кількість кадрів в секунду та надіслані команди. У перспективі доцільним також є зберігання швидкості руху та дані аналізу зображень.

Отримані дані групуються у сесії. Сесія являє собою єдину сутність, що об’єднує у собі усі дані, отримані в єдиний період часу, наприклад, під час дослідження однієї кімнати.

Основним актором потоків даних є клієнт, що посилає команди роботи з сесією, команди керування та запити на отримання відео та іншої інформації.

Потоки даних можна умовно поділити на чотири контури – контур відеоданих, контур команд керування, контур кількості кадрів в секунду та контур сесій.

Контур сесій відповідає за абстраговане збереження даних. Усі дані, отримані мобільною платформою у ході роботи зберігаються у сесії. Для підвищення швидкодії сесія зберігається в оперативній пам’яті. Після закінчення дослідження території сесію можна зберегти. Завдяки цій дії дані зберігаються в локальній реляційній базі даних. Клієнт може вивести список усіх сесій, де він бачить усю необхідну інформацію для аналізу усіх сесій дослідження території.

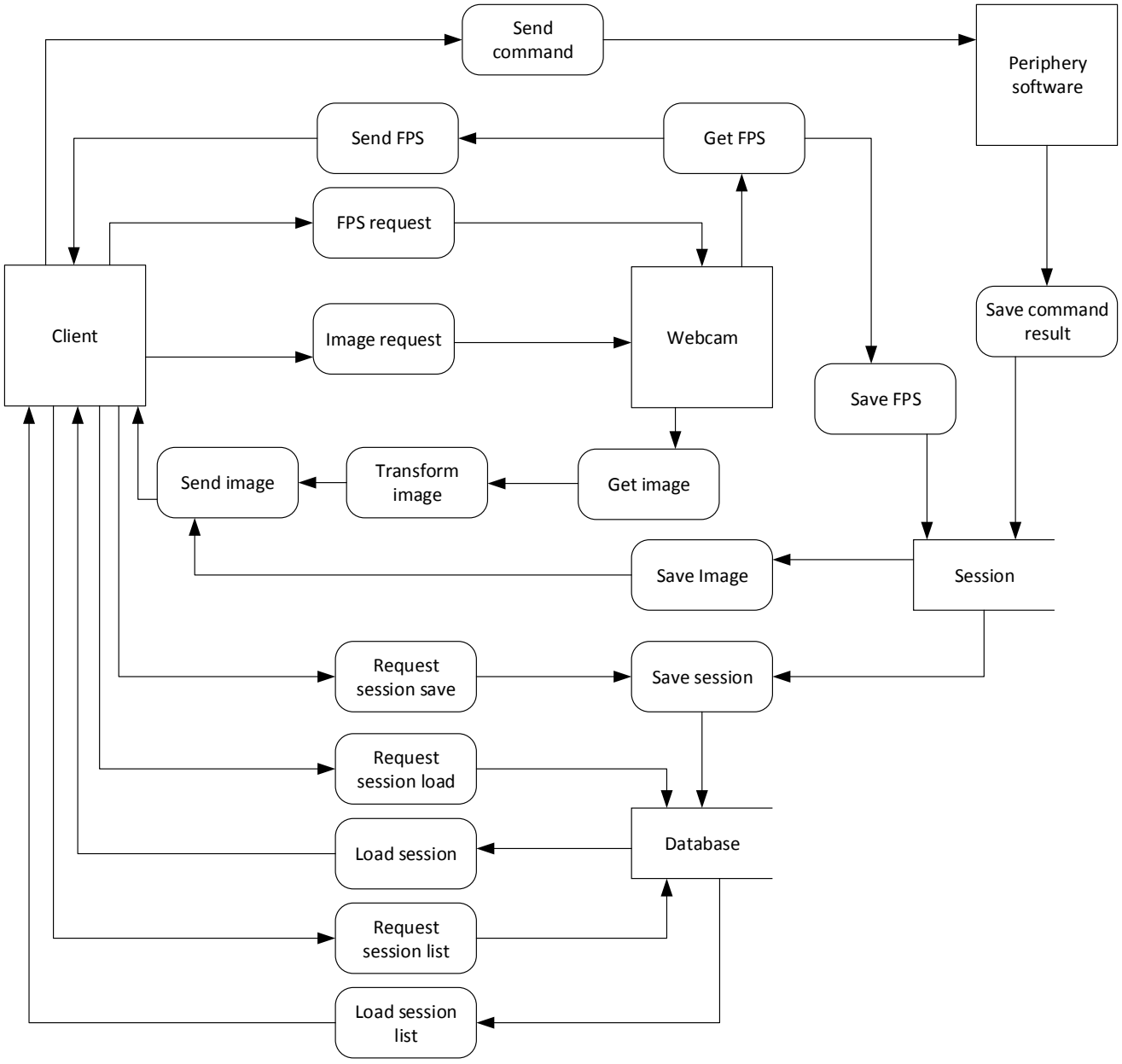

Рисунок 4.4 – Діаграма потоку даних підсистеми зберігання даних.

Якщо клієнт хоче дізнатись детальну інформацію щодо конкретної сесії, він може завантажити її із списку, і переглянути дані, отримані під час дослідження території, та результати аналізу, такі як карта місцевості.

У контуру відеоданих клієнт посилає запит на отримання зображення. Камера отримує запит, надсилає відповідь у вигляді зображення. З однієї сторони, зображення конвертується та надсилається до клієнта. З іншої, після конвертації зображення зберігається, локально або у файлосховищі. В будь якому випадку посилання на зображення, разом із часом отримання, зберігаються у сесії.

Контур команд керування відповідає за збереження команд керування для подальшого аналізу співвідношення між рухом та отриманими відеоданими Спочатку клієнт надсилає команду. Після обробки та перевірки зв’язку команда надсилається до програми керування двигунами. Відповідь щодо успішності виконання команди, разом із часом, надсилаються до сесії.

Контур збереження кількості кадрів в секунду відповідає за збереження інформації щодо здатності камери обробляти відеодані. Це дозволяє робити висновки щодо освітленості приміщення, швидкодії програмної інфраструктури та продуктивності програми. Клієнт періодично надсилає запит на отримання даних кількості кадрів в секунду, після чого дані надсилаються клієнту та записуються в сесію.

4.5 Висновки

У цьому розділі було розглянуто процес розробки програмної інфраструктури мобільної платформи для дослідження території. Була розроблена система передачі команд керування, беручи до уваги вимоги безпеки використання. Була розроблена та оптимізована система передачі поточного відео. Була розглянута і теоретично розроблена система збереження даних, визначені основні потоки даних та сценарії їх обробки.

Було проведене інтеграційне тестування. Результати інтеграційного тестування показали, що програма налаштована і працює коректно та очікувано.

ВИСНОВКИ

В ході виконання дипломної роботи було проведено дослідження предметної області, виділено головні ролі системи та бізнес-процеси. Проаналізовано вимоги до системи в цілому, вимоги до функцій системи, програмного і технічного забезпечення.

Було проведено дослідження технологій для побудови апаратної інфраструктури наземних роботів. В результаті дослідження були обрані наступні компоненти: плата Orange Pi Zero Plus, модуль керування двигунами L9110 H-bridge 24, двигуни постійного струму 6В, АА-батарейки та літій-іонний акумулятор серії 18650, і 5-мегапіксельна RPI Camera D.

Було побудовано повнофункціональний прототип системи на основі розглянутих компонентів та доведено роботоспроможність систем передачі команд, передачі потокового відео та збереження даних.

Було проведено дослідження технологій для побудови розподілених додатків. В результаті дослідження був обраний наступний стек засобів: сервер Apache Tomcat, Spring framework, Spring boot, JPA Hibernate. Ретельний огляд найпоширеніших СУБД дозволив обрати SqlLite та налаштувати її взаємодію з серверною частиною додатку.

Використання принципу тришарової архітектури додатку, тобто поділу на рівень представлення, рівень бізнес-логіки та рівень данних дало можливість розробити гнучку та ефективну систему, адже на кожному шарі відбувається вирішення окремих локальних задач, що позитивно відображається на надійності системи.

Результатом проведеного дослідження стало створення прототипу НРДТ. Система є сучасною та зручною, задовольняє всі вимоги з точки зору функціональності, юзабіліті, дизайну та безпеки даних.

ПЕРЕЛІК ІНФОРМАЦІЙНИХ ДЖЕРЕЛ ТА ПОСИЛАННЯ

1. Что такое микроконтроллеры - назначение, устройство, софт [електронний ресурс] — Режим доступу: http://electrik.info/main/automation/549-chto-takoe-mikrokontrollery-naznachenie-ustroystvo-princip-raboty-soft.html
2. Heath, Steve (2003). Embedded systems design. EDN series for design engineers (2 ed.). Newnes. pp. 11–12. ISBN 9780750655460.
3. The Single Board Computer Database [електронний ресурс] — Режим доступу: https://www.hackerboards.com/home.php
4. Best Single Board Computers 2019 [електронний ресурс] — Режим доступу: https://all3dp.com/1/single-board-computer-raspberry-pi-alternative/
5. What’s Orange Pi Zero Plus? [електронний ресурс] — Режим доступу: http://www.orangepi.org/OrangePiZeroPlus/
6. User Manual for Orange Pi [електронний ресурс] — Режим доступу: http://www.orangepi.org/downloadresources/orangepizeroplus/2017-08-15/orangepizeroplus3e9912e08037d6cdfc28.html
7. Зовнішній вигляд плати мікроконтролера Orange Pi Zero Plus [електронний ресурс] — Режим доступу: http://www.orangepi.org/images/pi_Zero%20plus_shuoming_en.jpg
8. Схема GPIO з фізичною нумерацією контактів [електронний ресурс] — Режим доступу: https://i.stack.imgur.com/O03j0.jpg
9. 1.3-мегапіксельна камера Gear Head WC740i-CP10 [електронний ресурс] — Режим доступу: https://images-na.ssl-images-amazon.com/images/I/41j7yZ-2yUL.jpg
10. Модуль L298N H-bridge 24 [електронний ресурс] — Режим доступу: https://arduino.ua/products_pictures/large_L298moduleRed4.jpg
11. L298N H-bridge datasheet [електронний ресурс] — Режим доступу: https://www.sparkfun.com/datasheets/Robotics/L298_H_Bridge.pdf
12. Banana Robotics: How to use the HG7881 (L9110) Dual Channel Motor Driver Module. – Режим доступа: https://www.bananarobotics.com/shop/How-to-use-

the-HG7881-(L9110)- Dual-Channel-Motor-Driver-Module. Дата доступа: 22.04.2016.

13. Загальний вигляд шасі у розібраному вигляді [електронний ресурс] — Режим доступу: https://arduino.ua/products_pictures/large_ARC100-2.jpg
14. Двигун постійного струму [електронний ресурс] — Режим доступу: https://arduino.ua/products_pictures/large_ARC121-1.jpg
15. Batteries—choose the right power source for your robot [електронний ресурс] — Режим доступу: https://medium.com/husarion-blog/batteries-choose-the-right-power-source-for-your-robot-5417a3ec19ca
16. 18650 Lithium Battery Charge Shield V3 [електронний ресурс] — Режим доступу: https://ae01.alicdn.com/kf/HTB1XTXEhlHH8KJjy0Fbq6AqlpXaY/18650-Lithium-Battery-Charge-Shield-V3-for-Raspberry-Pi-WEMOS-ESP32-De-board-for-Arduino-Micro.jpg_640x640.jpg
17. Types of batteries used in robotics [електронний ресурс] — Режим доступу: https://medium.com/husarion-blog/batteries-choose-the-right-power-source-for-your-robot-5417a3ec19ca
18. З'єднання пластикових частин системи за допомогою болтів та гайок [електронний ресурс] — Режим доступу: https://arduino.ua/products_pictures/large_ARC100-4.jpg
19. SD Memory Card Formatter for Windows [електронний ресурс] — Режим доступу: https://www.sdcard.org/downloads/formatter_4/eula_windows/
20. OrangePi download resources [електронний ресурс] — Режим доступу: http://www.orangepi.org/downloadresources
21. Win32 Disk Imager [електронний ресурс] — Режим доступу: http://sourceforge.net/projects/win32diskimager/files/Archive/
22. Spring Framework Overview [електронний ресурс]: https://docs.spring.io/spring/docs/current/spring-framework-reference/overview.html.

23. A java package to access the Capture interface of the Video4Linux API [електронний ресурс]: https://code.google.com/archive/p/v4l4j/
24. Wiring Pi GPIO Interface library for the Raspberry Pi [електронний ресурс]: http://wiringpi.com/
25. About SQLite [електронний ресурс]: https://www.sqlite.org/about.html
26. Hibernate ORM 5.4.2.Final User Guide [електронний ресурс]: http://docs.jboss.org/hibernate/orm/5.4/userguide/html_single/Hibernate_User_Guide.html
27. Hibernate ORM What is Object/Relational Mapping? [електронний ресурс]: https://hibernate.org/orm/what-is-an-orm/
28. TensorFlow Guide [електронний ресурс]: https://www.tensorflow.org/guide#ml_concepts
29. Tensorflow detection model zoo [електронний ресурс]: https://github.com/tensorflow/models/blob/master/research/object_detection/g3doc/detection_model_zoo.md

ДОДАТОК А

Код програми

```
package donkeybug.config;

import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.ComponentScan;
import org.springframework.context.annotation.Configuration;
import org.springframework.web.servlet.config.annotation.WebMvcConfigurer;
import org.springframework.web.servlet.view.InternalResourceViewResolver;
import org.springframework.web.servlet.view.JstlView;

@Configuration
@ComponentScan(basePackages = {"donkeybug.*"})
public class AppConfig implements WebMvcConfigurer {
    @Bean
    public InternalResourceViewResolver viewResolver() {
        InternalResourceViewResolver viewResolver = new
InternalResourceViewResolver();
        viewResolver.setViewClass(JstlView.class);
        viewResolver.setPrefix("/WEB-INF/jsp/");
        viewResolver.setSuffix(".jsp");
        return viewResolver;
    }
}

package donkeybug.config;

import org.springframework.context.annotation.Configuration;
import org.springframework.messaging.simp.config.MessageBrokerRegistry;
import
org.springframework.web.socket.config.annotation.EnableWebSocketMessageBroker;
import org.springframework.web.socket.config.annotation.StompEndpointRegistry;
import
org.springframework.web.socket.config.annotation.WebSocketMessageBrokerConfigure
r;

@Configuration
@EnableWebSocketMessageBroker
public class WebSocketConfig implements WebSocketMessageBrokerConfigurer {

    @Override
    public void configureMessageBroker(MessageBrokerRegistry config) {
        config.enableSimpleBroker("/topic");
        config.setApplicationDestinationPrefixes("/app");
    }

    @Override
    public void registerStompEndpoints(StompEndpointRegistry registry) {
        registry.addEndpoint("/donkeybug_websocket").withSockJS();
    }

}

package donkeybug.controller;

import donkeybug.service.car.CarService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.messaging.handler.annotation.MessageMapping;
import org.springframework.stereotype.Controller;
```

```java
@Controller
public class CommandController {
   @Autowired
   private CarService carService;

   @MessageMapping("/command")
   public void command(String command) {
      switch (command) {
         case "stop":
            carService.stop();
            break;
         case "forward":
            carService.goForward();
            break;
         case "left":
            carService.turnLeft();
            break;
         case "right":
            carService.turnRight();
            break;
         case "back":
            carService.goBackward();
            break;
         default:
            carService.stop();
      }
   }
}
```

```java
package donkeybug.controller;

import donkeybug.service.car.CarService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.messaging.handler.annotation.MessageMapping;
import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.GetMapping;

@Controller
public class HomeController {
    @GetMapping("/index")
    public String index() {
        return "index";
    }
}
```

```java
package donkeybug.controller;

import donkeybug.service.webcam.WebcamService;
import donkeybug.service.webcam.WebcamViewer;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.beans.factory.annotation.Qualifier;
import org.springframework.http.MediaType;
import org.springframework.messaging.simp.SimpMessagingTemplate;
import org.springframework.scheduling.annotation.EnableScheduling;
import org.springframework.scheduling.annotation.Scheduled;
import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.ResponseBody;

import javax.annotation.PostConstruct;
import javax.imageio.ImageIO;
import java.awt.image.BufferedImage;
```

```
import java.io.ByteArrayOutputStream;
import java.io.IOException;
import java.util.Optional;

@EnableScheduling
@Controller
public class WebcamController {
   @Autowired
   private WebcamViewer webcamViewer;

    @Autowired
    private SimpMessagingTemplate template;

   @GetMapping(value = "/webcam", produces = MediaType.IMAGE_JPEG_VALUE)
   public @ResponseBody byte[] getFeed() {
      byte[] byteArray = webcamViewer.getRawView();
        return byteArray;
   }

   @Scheduled(fixedRate = 1000)
   public void sendFPS()  throws Exception {
      template.convertAndSend("/topic/fps", (int)
Math.round(webcamViewer.getFPS()));
   }
}

package donkeybug.service.car;

import org.springframework.beans.factory.annotation.Value;
import org.springframework.stereotype.Service;

import java.io.*;
import java.util.concurrent.BrokenBarrierException;
import java.util.concurrent.CyclicBarrier;

@Service
public class CarConsoleManager implements ConsoleManager {
    @Value("${OS}")
    private String OS;

    private CyclicBarrier responseBarrier;

    private Process process;

    private PrintWriter writer;

    private StreamGobbler sgInput;
    private StreamGobbler sgError;

    private String response;

    @Override
    public void startApp() {
        try {
            ProcessBuilder builder;
            switch (OS) {
                case "Linux":
                    builder = new ProcessBuilder("sudo", "periphery/car");
                    break;
                case "Windows":
                    builder = new ProcessBuilder("periphery/car.exe");
                    break;
                default:
```

```
                builder = new ProcessBuilder("periphery/car.exe");
            }

            Process process = builder.start();

            // create the stream gobblers, one for the input stream and one for
the
            // error stream. these gobblers will consume these streams.
            sgInput = new StreamGobbler(
                    process.getInputStream(), "input");
            sgError = new StreamGobbler(
                    process.getErrorStream(), "error");

            responseBarrier = new CyclicBarrier(2);

            Thread inputThread = new Thread(sgInput);
            inputThread.start();
            Thread errorThread = new Thread(sgError);
            errorThread.start();

            writer = new PrintWriter(process.getOutputStream());
        } catch (Exception e) {
            e.printStackTrace();
        }
    }

    @Override
    public String sendCommand(String command) {
        responseBarrier = new CyclicBarrier(2);

        // sends the command to the process
        // simulating an user input (note the \n)
        writer.write(command);
        writer.write("\n");
        writer.flush();

        try {
            responseBarrier.await();
        } catch (InterruptedException ex) {
            System.out.println(ex.toString());
            return ex.toString();
        } catch (BrokenBarrierException ex) {
            System.out.println(ex.toString());
            return ex.toString();
        }
        return response;
    }

    @Override
    public void closeApp() {
        sendCommand("end");
        process.destroy();
    }

    private class StreamGobbler implements Runnable {

        private InputStream is;
        private String type;
        private FileWriter fw;

        public StreamGobbler(InputStream is, String type) {
            this.is = is;
            this.type = type;
```

```
        }

        public StreamGobbler(InputStream is, String type, File file)
                throws IOException {
            this.is = is;
            this.type = type;
            this.fw = new FileWriter(file);
        }

        @Override
        public void run() {
            try {
                InputStreamReader isr = new InputStreamReader(is);
                BufferedReader br = new BufferedReader(isr);
                String line = null;
                while ((line = br.readLine()) != null) {
                    if (fw != null) {
                        fw.write(line + "\n");
                    } else {
                        response = type + ">" + line;
                        System.out.println(response);
                    }

                    try {
                        responseBarrier.await();
                    } catch (InterruptedException ex) {
                        return;
                    } catch (BrokenBarrierException ex) {
                        return;
                    }
                }
                if (fw != null) {
                    fw.close();
                }
            } catch (IOException ioe) {
                ioe.printStackTrace();
            }
        }
    }
}

package donkeybug.service.car;

public interface CarService {
    void goForward();

    void goBackward();

    void turnLeft();

    void turnRight();

    void stop();
}

package donkeybug.service.car;

public interface ConsoleManager {
    void startApp();

    String sendCommand(String command);
```

```
    void closeApp();
}
```

```
package donkeybug.service.car;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

import javax.annotation.PostConstruct;
import javax.annotation.PreDestroy;
import java.util.Observable;

@Service
public class ContinuousCarService extends Observable implements CarService {
    @Autowired
    private ConsoleManager consoleManager;

    private String command;
    private boolean receivedCommand;
    private CommandSender commandSender;

    @PostConstruct
    public void initIt() throws Exception {
        consoleManager.startApp();

        command = "q";
        receivedCommand = false;
        consoleManager.sendCommand("q");

        commandSender = new CommandSender();
        Thread CommandThread = new Thread(commandSender);
        CommandThread.start();
    }

    @Override
    public synchronized void goForward() {
        receiveCommand("w");
    }

    @Override
    public synchronized void goBackward() {
        receiveCommand("s");
    }

    @Override
    public synchronized void turnLeft() {
        receiveCommand("a");
    }

    @Override
    public synchronized void turnRight() {
        receiveCommand("d");
    }

    @Override
    public synchronized void stop() {
        receiveCommand("q");
    }

    @PreDestroy
    public void cleanUp() throws Exception {
        consoleManager.closeApp();
    }
```

```
    private void receiveCommand(String command) {
        this.command = command;
        receivedCommand = true;
        if (commandSender != null) {
            synchronized (commandSender) {
                commandSender.notify();
            }
        }
    }

    private class CommandSender implements Runnable {
        String previousCommand;

        @Override
        public synchronized void run() {
            while (true) {
                previousCommand = command;
                receivedCommand = false;

                try {
                    this.wait(200);
                } catch (InterruptedException e) {
                    e.printStackTrace();
                }

                if (!receivedCommand) {
                    command = "q";
                    if (!previousCommand.equals(command)) {
                        consoleManager.sendCommand(command);
                    }
                    previousCommand = command;
                } else if (!previousCommand.equals(command)) {
                    consoleManager.sendCommand(command);
                }
            }
        }
    }
}
```

```
package donkeybug.service.webcam;

import com.github.sarxos.webcam.Webcam;
import com.github.sarxos.webcam.ds.v4l4j.V4l4jDriver;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.beans.factory.annotation.Value;
import org.springframework.stereotype.Service;

import javax.annotation.PostConstruct;
import javax.annotation.PreDestroy;
import javax.imageio.ImageIO;
import java.awt.*;
import java.awt.image.BufferedImage;
import java.io.ByteArrayOutputStream;
import java.io.IOException;
import java.util.Optional;

@Service
public class SimpleWebcamViewer implements WebcamViewer {
    @Autowired
    WebcamService webcamService;

    @Value("${OS}")
```

```
    String OS;

    byte[] toByte(BufferedImage bufferedImage) {
        try {
            ByteArrayOutputStream baos = new ByteArrayOutputStream();
            ImageIO.write(bufferedImage, "jpeg", baos);
            byte[] byteArray = baos.toByteArray();

            return byteArray;
        } catch (IOException e) {
            e.printStackTrace();
            return null;
        }
    }

    @Override
    public BufferedImage getImage() {
        Optional<Webcam> webcam = webcamService.getWebcam();
        BufferedImage result = webcam.map(Webcam::getImage)
                .orElse(new BufferedImage(1,1, BufferedImage.TYPE_INT_RGB));
        return result;
    }

    @Override
    public double getFPS() {
        Optional<Webcam> webcam = webcamService.getWebcam();
        double result = webcam.map(Webcam::getFPS)
                .orElse(0.0);
        return result;
    }

    @Override
    public byte[] getRawView() {
        return toByte(getImage());
    }
}
```

```
package donkeybug.service.webcam;

import com.github.sarxos.webcam.Webcam;
import com.github.sarxos.webcam.ds.v4l4j.V4l4jDriver;
import org.springframework.beans.factory.annotation.Value;
import org.springframework.stereotype.Service;

import javax.annotation.PostConstruct;
import javax.annotation.PreDestroy;
import java.awt.*;
import java.awt.image.BufferedImage;
import java.util.Optional;

@Service
public class WebcamCaptureWebcamService implements WebcamService{
    @Value("${OS}")
    private String OS;

    private Webcam webcam;

    @PostConstruct
    public void initIt() throws Exception {
        switch (OS) {
            case "Linux":
                Webcam.setDriver(new V4l4jDriver()); // this is important (not
really)
```

```
                break;
            }
            // get default webcam and open it
            webcam = Webcam.getDefault();
            if (webcam != null) {
                webcam.setViewSize(new Dimension(320, 240));
                webcam.open(true);
            }
        }

        @Override
        public Optional<Webcam> getWebcam() {
            return (webcam != null && webcam.isOpen()) ? Optional.of(webcam) :
Optional.empty();
        }

        @PreDestroy
        public void cleanUp() throws Exception {
            webcam.close();
        }
}
```

```
package donkeybug.service.webcam;

import com.github.sarxos.webcam.Webcam;

import java.awt.image.BufferedImage;
import java.util.Optional;

public interface WebcamService {
    Optional<Webcam> getWebcam();
}
```

```
package donkeybug.service.webcam;

import javax.imageio.ImageIO;
import java.awt.image.BufferedImage;
import java.io.ByteArrayOutputStream;
import java.io.IOException;
import java.util.Optional;

public interface WebcamViewer {
   byte[] getRawView();
   BufferedImage getImage();
   double getFPS();
}
```

```
package donkeybug;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import
org.springframework.boot.web.servlet.support.SpringBootServletInitializer;

@SpringBootApplication
public class DonkeybugApplication extends SpringBootServletInitializer {

    public static void main(String[] args) throws Exception {
        SpringApplication.run(DonkeybugApplication.class, args);
    }
}
```

```css
* {
   box-sizing: border-box;
}

body {
   background-position: center;
   background-size: contain;
   background-repeat: no-repeat;
}

.hud {
   height: 25%;
}

.moveBlock {
   border-radius: 2vw;

   position: fixed;
   top: 50%;
   left: 80%;
   width: 15vw;
   height: 20vw;
   background-color: #BCF0FC;
   border: 3px solid #244A9D;

   display: flex;
   flex-direction: column;
   justify-content: space-evenly;
}

.moveButton {
   display: flex;
   flex-direction: row;
   justify-content: center;
}

.rotateBlock {
   border-radius: 2vw;
   position: fixed;
   top: 60%;
   right: 75%;
   width: 20vw;
   height: 15vw;
   background-color: #BCF0FC;
   border: 3px solid #244A9D;

   display: flex;
   flex-direction: row;
   justify-content: space-evenly;
}

.rotateButton {
   display: flex;
   flex-direction: column;
   justify-content: center;
}

.arrow {
   font-size:8vw;
   outline: none;
   background-color: #BCF0FC;
    transition-duration: 0.4s;
    border: none;
```

```
   color:#244A9D;
}

.arrow:hover {
    color: grey;
    text-shadow: 2px 2px 4px #000000;
}

.arrow:active{
    color: white;
}

.imageContainer {
   position: fixed;
   height: 100%;
   width: 100%;
   object-fit: contain;
}

html {
   font-family: "Lucida Sans", sans-serif;
}

var stompClient = null;

var socketUrl = '/donkeybug_websocket';

var command = "stop";

var feedReady = true;

function connect() {
   var socket = new SockJS(socketUrl);
   stompClient = Stomp.over(socket);
   stompClient.connect({}, function (frame) {
      console.log('Connected: ' + frame);
      stompClient.subscribe('/topic/fps', function (fps) {
                showFPS(JSON.parse(fps.body));
            });
      var commandTimer = setTimeout(function run() {
            stompClient.send("/app/command", {}, command);
            setTimeout(run, 100);
        }, 100);
        var webcamTimer = setInterval(function run() {
            if (feedReady) {
                feedReady = false;
                getImage();
            }
        }, 33);
   });
}

function reconnect(socketUrl) {
    let reconInv = setInterval(() => {
        connect();
        clearInterval(reconInv);
    }, 100);
}

function disconnect() {
   if (stompClient !== null) {
      stompClient.disconnect();
   }
```

```
    //setConnected(false);
    console.log("Disconnected");
    reconnect(socketUrl);
}

function showFPS(fps) {
    $('#fps').text("FPS: " + fps);
    console.log('RECEIVED FPS: ' + fps);
}

function getImage() {
    var oReq = new XMLHttpRequest();
    oReq.open("GET", "/webcam", true);
    oReq.responseType = "arraybuffer";

    oReq.onload = function(oEvent) {
        var blob = new Blob([oReq.response], {type: "image/jpeg"});

        var urlCreator = window.URL || window.webkitURL;
        var imageUrl = urlCreator.createObjectURL(blob);
        document.querySelector("#WebcamFeed").src = imageUrl;
        urlCreator.revokeObjectURL(blob);
        feedReady = true;
    };
    oReq.send();
}

$(function () {

    connect();
    $( "#forward" ).on('touchstart mousedown', function(e) {
        e.preventDefault();
        //sendCommand("forward");
        command = "forward";
    });
    $( "#forward" ).on('touchend mouseup', function() {
        //sendCommand("stop");
        command = "stop";
    });

    $( "#left" ).on('touchstart mousedown', function(e) {
        e.preventDefault();
        //sendCommand("left");
        command = "left";
    });
    $( "#left" ).on('touchend mouseup', function() {
        //sendCommand("stop");
        command = "stop";
    });

    $( "#stop" ).on('touchstart mousedown', function(e) {
        e.preventDefault();
        //sendCommand("stop");
        command = "stop";
    });

    $( "#right" ).on('touchstart mousedown', function(e) {
        e.preventDefault();
        //sendCommand("right");
        command = "right";
    });
    $( "#right" ).on('touchend mouseup', function() {
        //sendCommand("stop");
```

```
        command = "stop";
    });

    $( "#back" ).on('touchstart mousedown', function(e) {
        e.preventDefault();
        //sendCommand("back");
        command = "back";
    });
    $( "#back" ).on('touchend mouseup', function() {
        //sendCommand("stop");
        command = "stop";
    });
});

<%@ page language="java" contentType="text/html; charset=UTF-8"
   pageEncoding="UTF-8"%>
<%@ taglib prefix="spring" uri="http://www.springframework.org/tags"%>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core"%>

<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">

<html lang="en">
<head>
   <!-- Styles -->
   <link href="/webjars/bootstrap/css/bootstrap.min.css" rel="stylesheet">
    <link href="css/main.css" rel="stylesheet">
    <link rel='stylesheet'
href='https://use.fontawesome.com/releases/v5.5.0/css/all.css'
integrity='sha384-
B4dIYHKNBt8Bc12p+WXckhzcICo0wtJAoU8YZTY5qE0Id1GSseTk6S+L3BlXeVIU'
crossorigin='anonymous'>

    <!-- Scripts -->
    <script src="/webjars/jquery/jquery.min.js"></script>
    <script src="/webjars/sockjs-client/sockjs.min.js"></script>
    <script src="/webjars/stomp-websocket/stomp.min.js"></script>
    <script src="js/app.js"></script>
</head>
<body>
    <img class="imageContainer" id="WebcamFeed" src="" />

    <div class="fps">
        <p id="fps"></p>
    </div>

    <div class="moveBlock">
        <div class="moveButton">
            <button id="forward" class='fas fa-arrow-up arrow'></button>
        </div>
        <div class="moveButton">
            <button id="back" class='fas fa-arrow-down arrow'></button>
        </div>
    </div>

    <div class="rotateBlock">
        <div class="rotateButton">
            <button id="left" class='fas fa-arrow-left arrow'></button>
        </div>
        <div class="rotateButton">
            <button id="right" class='fas fa-arrow-right arrow'></button>
        </div>
    </div>
```

```
</body>

</html>
```